

· 代码逆袭 ·

超实用的 CSS 代码段

赵荣娇 任建智 编著

電子工業出版社·

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书精选 400 余段 CSS 代码，覆盖网页上所有的设计元素，堪称史上最有用的 CSS 书籍，是网站建设和网页设计人员不可或缺的解决方案、技巧和模板。本书的代码跨平台、跨设备、跨浏览器，充分向读者演示了如何使用 CSS 的各项技术，实现令人炫目的网页布局和效果。

本书从网页效果的不同类型和使用场景，对常用的 CSS 代码段进行了全方位的介绍和演示。全书分为 11 章，包含文字、字体、边框、图片、按钮、链接、背景、颜色、动画、页面布局、美化、盒子、3D、CSS Hack 等网页设计和交互技术，对那些客户要求高、工作节奏快的网站开发人员和设计人员有着尤其重要的指导作用。

本书内容简洁明了、代码精练、重点突出、实例丰富、语言通俗易懂、原理清晰明白，是广大网页设计入门者和提高者的良好选择，同时也非常适合大中专院校学生学习阅读，也可作为高等院校非计算机专业，以及计算机非网络工程和相关专业的辅助读物。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

超实用的 CSS 代码段 / 赵荣娇，任建智编著. —北京：电子工业出版社，2014.9
（代码逆袭）
ISBN 978-7-121-23948-9

I. ①超… II. ①赵… ②任… III. ①网页制作工具 IV. ①TP393.092

中国版本图书馆 CIP 数据核字（2014）第 173085 号

责任编辑：董 英

印 刷：北京京科印刷有限公司

装 订：北省三河市路通装订厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×1092 1/16 印张：24.25 字数：602 千字

版 次：2014 年 9 月第 1 版

印 次：2014 年 9 月第 1 次印刷

定 价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlls@phei.com.cn，盗版侵权举报请发邮件到 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言 →

CSS 是网站一件美丽的外衣，没有 CSS，我们的网页不会如此丰富多彩。所以，要搭建网站和设计网页，就必须要学会 CSS。本书是一本研究代码实践的书，它为读者全面深入地讲解了针对各种屏幕大小设计和开发现代网站的 CSS 技术。400 多段代码给读者带来的不仅仅是网页设计的提速，更是教会读者如何应对跨浏览器兼容，如何处理语义化、无障碍访问、搜索引擎优化、创建高性能网页等时时刻刻困扰网页设计人员的问题。

CSS 中的那些槛儿

有些网页设计人员做了多年 CSS，依然在面对问题时束手无策，下列问题很常见，你又能了解多少呢？

- 浏览器的兼容
- 盒子模型
- 绝对定位和相对定位
- 流和浮动
- 自适应的文字、DIV、图片
- Webkit 内核的浏览器的特殊属性
- 动态宽度的布局和固定宽度的布局差异
- IE 6 的兼容

以上所有内容在本书的代码中都有讲解，除这些常见 CSS 门槛外，本书力求将最有用的 CSS 代码汇总在一起，提供各种解决实际问题的跨浏览器方案。

如何学习 CSS

11 个字就能帮助我们更好地学习 CSS。

- **多看、多练：**观摩成功的网页设计，分析并练习网页设计常用的代码。
- **多想、多问：**思考设计实现的原理，提出自己的问题并通过各种渠道来找答案。
- **多总结：**记录前人已经探索出来的 CSS 技巧，总结实战中碰到的问题及解决方案。

只要真正能做到勤思考、勤动手、勤总结，CSS 学习定能一马平川。

本书的编写特点

1. 独特的 CSS 切入点

与市面上其他 CSS 有关的书不同，本书从最常见的网页效果出发，直接应用 CSS 代码实现设计效果，没有用的例子不给，只讲事实不讲废话！

2. 内容丰富，知识全面

本书以网页设计的各个分类和使用场景作为基础，立体式全方位地解释各种场景下的 CSS 代码段应用，场景丰富、实例丰富，并拥有良好的可扩展性、可复用性。

3. 去中心化，分布式学习

本书的代码实例都是独立的，你可以从中间开始学，也可以从头开始学。代码跨平台跨设备，你可以在 iPad 上学，也可以在 PC 上学，如果可以，手机上学代码也完全可能。

4. 解释清晰，原理结合实践

由于 CSS 是网页描述性语言，虽然语法简单，但是很多读者可能不知效果从何做起，本书通过清晰的实现原理分析，配备简单易懂的代码和解释，从效果的实现原理方面进行了剖析，使读者不仅能知其然，更知其所以然。

5. 多种代码方式的实现

本书的实例从纯 CSS 代码、简单的原生 JavaScript 配合、jQuery 框架和自己搭建框架等多种方式实现不同的效果，配合原理的说明，可在不同的方式间自由切换。

6. 自发式学习

在学习代码前，先让读者练习实际上最基础却最容易做错的 CSS 考题和面试题，激发读者的学习斗志。

本书的设计始于功能、终于代码，是网页设计人员的案头必备。

本书的内容安排

本书共 11 章，各章节为不同类别效果的 CSS 实现。

第 1 章为文字与字体，介绍常用的网页文本样式的设计，包括自定义字体、文本缩进与首字符下沉、文本对齐、文字间距、文本溢出、文字阴影、毛玻璃效果、金属质感等装饰、隐藏文本、文字旋转、现代字体栈等。

第 2 章为边框和图片，介绍网页中常见的边框设计和各种图片的展示方式，包括各种边框、黑白图片、图片水印、细节放大、瀑布流、图片墙、图片轮播图（焦点图）、幻灯片（带缩略图）、图片自适应、图片原地放大、图片翻转、图像地图等。

第 3 章为按钮和链接，介绍常见的网页按钮和链接相关的设计，包括圆角按钮、导航菜单、下拉菜单、右键菜单、标签云、文字分享、iPhone 开关、按钮式单选框与复选框、自定义播放器、文字变链接等。

第4章为背景和颜色，介绍网页中背景和颜色的使用，包括高光效果页面顶部阴影、多背景、全屏背景、斑马线背景、棋盘背景、易拉罐效果等。

第5章为变换与动画的相关内容，是一些网页中动态效果的集合，包括纸张边角动画、气泡式提示、对联广告、页面 loading 效果、进度条、模拟时钟、苹果系统的 DOCK 栏和 Stack 特效、扇形展开等。

第6章介绍页面的布局，包括图文混排、几种不同的居中布局方法、绝对定位与相对定位、适配 iPad 屏幕的布局、Clearfix、滚动条的控制、CSS 3 文本分列、Metro 和 Flexbox 布局风格等。

第7章为美化与装饰，学习如何在细节方面美化网页，包括文本的装饰、锚链接装饰、自定义滚动条、分隔线、列表符号、跨浏览器的透明度、鼠标指向特效、翻页页码、顶部阴影、页面卷曲、针线缝合效果等。

第8章主要介绍 CSS 中的盒子效果，包括 CSS 3 盒模型、内层 CSS 3 盒阴影、外层 CSS 3 盒阴影、纯 CSS 3 透明水晶盒、投影发光效果。

第9章内容与 3D 相关，包含 3D 文字、3D 图片立体效果、3D 按钮、3D 下拉菜单和 3D 旋转动画，使网页更加立体化。

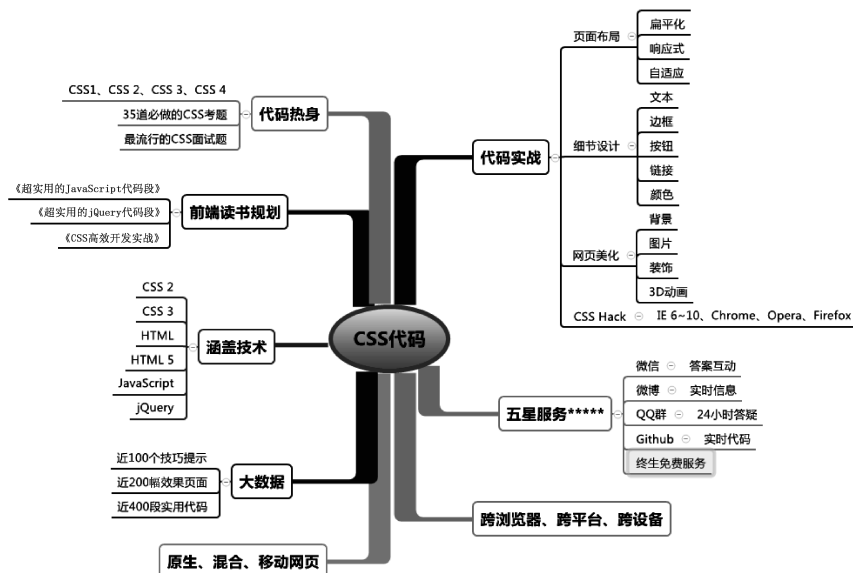
第10章内容为 CSS Hack，解决因不同浏览器性能的不同带来的效果不一致和效果出错的问题，包含让网站在所有浏览器下显示一致、解决 IE 6 的浮动元素的双倍边距问题、识别不同浏览器、背景与图片透明、IE10 CSS Hack、CSS 3 滤镜。

第11章为其他常用代码，既包含某些 CSS 代码技巧，也包含一些不便于分类的代码，还包含某些 JavaScript 与 CSS 结合实例中的 JavaScript 解决方案，内容包含 CSS 块引用模版、一般媒体查询与响应式设计、字符编码、手机 APP 设计等。

本书面对的读者

- 网页设计入门者
- 网页开发入门者
- CSS 学习爱好者
- 由 CSS 2 向 CSS 3 转型的开发人员
- 网页美工人员
- 中小型企业网站开发者
- 大中专院校的学生
- 各种 IT 培训学校的学生
- 网站后台开发人员
- 网站建设与网页设计的相关威客兼职人员

本书的思维导图



编者推荐

本书摒弃传统的说教模式，每段代码都是单独的功能型页面，读者可以从全书的任意一点开始线性阅读。本书的目的就是将最有用的代码与读者分享，包含了网页设计人员在实战中必须具备的所有技巧和方法，读者可以拿来就用。本书的 400 段代码也许并不是最优的代码，但笔者提供了 Github 地址，与世界 CSS 工程师一起优化这些代码，并实现了更新迭代，以保证读者始终能看到最好的、最高效的、最实用的 CSS 代码段。这是一本市场上绝无仅有的 CSS 实战书，是一本值得拥有的 CSS 设计书。

本书的服务

笔者能力有限，如果写作过程中有什么疏漏，或者读者有什么疑问，可通过以下方式与我们沟通。

- QQ 群：296811675，作者在线答疑。
- 扫描封底的微信二维码，时刻参与我们的图书互动和本书的考题答案。
- @博文视点的微博，了解我们发布的信息和各种前端流行技术。
- 博文视点官方网站 <http://www.broadview.com.cn/>，下载本书所有实例源代码。
- Github, <https://github.com/yinqiao/supercss>，了解代码的实时更新和迭代过程，并可以在每章代码下参与讨论，也可以观看其他读者提出的问题，还可以随时随地下载代码。

很多读者在学习过程中苦于无法交流，小故障无法及时解决，加入我们的服务方阵，我们将为您提供终身免费的服务。一本书、一段情、一辈子。

本书主要由赵荣娇、任建智编写，参与编写的人员还有李勇、周敏、王铁民、张兴瑜、马新原、薛淑英、殷龙、席新亮、谢郁、于健、周洋、李春城、李兰英。

序 1 CSS 的前世今生



本书的案例允许读者没有一丁点 CSS 基础，以实践代码为主，帮助读者解构网页中的每个元素，包括文本、链接、图片、层、边框、背景、颜色、布局、美化等所有你能在网页中看到的视觉元素。

这不是一本普通的 CSS 书，它涉及网页设计的技巧、网页性能的优化、拿来主义的应用。它不是你的老师，它只是你的左右手，拥有这些代码，你将会在前端工作中战无不胜！

CSS 1、CSS 2、CSS 3、CSS 4

CSS（Cascading Style Sheets），翻译为层叠样式表。运用 CSS，不仅可以设计出精美的页面效果，还能提高网站的可访问性、可维护性以及用户体验。CSS 已经被绝大多数浏览器使用的有 3 个版本：CSS 1、CSS 2 和 CSS 3。

CSS 1 提供有关字体、颜色、位置和文本属性的基本信息，该版本已经得到了目前解析 HTML 和 XML 的浏览器的广泛支持。CSS 1 还未与 HTML 结构脱离，即结构与样式混合嵌套。

CSS 2 推荐的是一套内容和表现效果分离的方式，HTML 元素可以通过 CSS 2 的样式控制显示效果，可完全不使用以往 HTML 中的 `table` 和 `td` 来定位表单的外观和样式，只需使用 `<div>` 和 `` 此类 HTML 标签来分割元素，之后即可通过 CSS 2 样式来定义表单界面的外观。

CSS 3 是 CSS 技术的升级版本，CSS 3 语言开发是朝着模块化发展的。以前的规范作为一个模块实在是太庞大而且比较复杂，所以，把它分解为一些小的模块，更多新的模块也被加入进来。这些模块包括盒子模型、列表模块、超链接方式、语言模块、背景和边框、文字特效、多栏布局、边框、阴影、过渡、3D 动画等。

CSS 的发展可以用表 1 来说明。

表 1 CSS 的发展

时间	特点
很久以前	浏览器不支持任何的 CSS 属性，所以它们的顺序并不重要
过去一段时间	只有带供应商前缀的属性被支持，所以这些顺序也不重要

续表

时间	特点
现在	带供应商前缀的属性和标准属性都被支持，所以顺序很重要，要把标准属性写在最后，以便让其在被支持时能覆盖供应商属性
将来	只有标准属性被支持，所以顺序又将变得不重要

W3C 于 2011 年 9 月 29 日开始设计 CSS 4，直至现时只有极少数的功能被部分网页浏览器支持，如使用在 HTML 而非 SVG 上的 `pointer-events`。CSS `pointer-events` 属性允许控制在一定情况下（如果有的话）一个特定的图形元素可以成为鼠标事件的目标，当这个属性未指明时，相同可见特征适用于 SVG 内容。CSS 4 增加了一些更方便的方法以选择不同元素，如：

```
01  $ul li.clicked {  
02      background: white;  
03  }
```

CSS 4 里一项新功能是能够定义于选择器链中哪一个是目标元素。在上一段代码中，`ul` 前面加了一个 `$` 符号，代表它就是要改变的目标，即把里面有 `li`（`class` 是 `clicked`）的 `ul` 的底色转为白色。

CSS 4 里另一项新功能就是能够使用 `:matches()` 来简化选择器，例如：

```
04  :matches(div, p, nav) span{  
05      font-size: 18px;  
06  }
```

此段代码等同于：

```
07  div span, p span, nav span{  
08      font-size: 18px;  
09  }
```

事实上现在已有浏览器支持类似的功能，如 Firefox 和 Webkit：

```
10  /*Firefox 使用-moz-前缀*/  
11  -moz-any(div, p, nav) span{  
12      font-size: 18px;  
13  }  
14  /*Webkit 使用-webkit-前缀*/  
15  -webkit-any(div, p, nav) span{  
16      font-size: 18px;  
17  }
```

尽管 CSS 4 的内容并不为浏览器所广泛支持，但是，CSS 的更高标准仍然在进一步的推广中。

CSS 的学习目标

在学习 CSS 之前，需要对 HTML、XHTML 有基本的了解。学习 CSS 的目标是如何使

用 CSS 同时控制多重网页的样式和布局，通过使用 CSS 来提升工作效率。

CSS 的学习目标是精通 HTML/XHTML、CSS，熟悉页面架构和布局，对 Web 标准和标签语义化有深入理解，对目前互联网流行的网页制作方法，以及各大浏览器兼容性有极大的了解。

本书的目的就是帮助你：

- 掌握 CSS 基础语法，包括 CSS 的引用方式、id 选择器、类选择器等，如：

```
selector {declaration1; declaration2; ... declarationN }
```

- 掌握 CSS 的引用方式，包括外部样式表、内部样式表、内联样式、多重样式等方式。
- 掌握 CSS 选择器、CSS 样式、CSS 框模型、CSS 定位等技巧。
- 掌握 CSS 高级语法的使用，包括选择器分组、派生选择器、后代选择器、子元素选择器、相邻兄弟选择器、属性选择器、继承及其他问题，如：

```
h1,h2,h3,h4,h5,h6 { color: green; }
```

```
body { font-family: Verdana, sans-serif; }
```

```
p, td, ul, ol, li, dl, dt, dd { font-family: Verdana, sans-serif; }
```

- 解决各种浏览器兼容性问题。
- 学习网站性能优化方法。
- 掌握前沿技术（HTML 5+CSS 3）的组合。

30 段必须拥有的 CSS 代码

在开始介绍本书实例前，笔者摘选了 30 段最通用的代码段，不解释，你值得拥有！这些代码穿插在本书的代码实例中，读者如果不太明白，可以通过目录找到具体实例亲自动手练习。

（1）清除浮动的方法

```
.clearfix:before, .container:after { content: ""; display: table; }
.clearfix:after { clear: both; }
/* IE 6/7 */
.clearfix { zoom: 1; }
```

（2）跨浏览器设置透明度

```
.transparent {
    filter: alpha(opacity=50); /* IE */
    -khtml-opacity: 0.5;      /* 老版本的 Safari */
    -moz-opacity: 0.5;        /* Mozilla, Netscape */
    opacity: 0.5;             /* fx, Safari, Opera */
}
```

（3）设置圆角

```
#container {
    -webkit-border-radius: 4px 3px 6px 10px;
```

```
-moz-border-radius: 4px 3px 6px 10px;
-o-border-radius: 4px 3px 6px 10px;
border-radius: 4px 3px 6px 10px;
}
```

(4) 一般媒体查询方法 (针对 PC、智能设备、大屏幕等)

```
/* 智能设备(横屏和竖屏) ----- */
@media only screen
and (min-device-width : 320px) and (max-device-width : 480px) {
    /* 样式设计 */
}
/* 智能设备(竖屏) ----- */
@media only screen and (min-width : 321px) {
    /* 样式设计 */
}
/*智能设备(横屏) ----- */
@media only screen and (max-width : 320px) {
    /* 样式设计 */
}
/* iPads (横屏和竖屏) ----- */
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) {
    /* 样式设计 */
}
/* iPads (竖屏) ----- */
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) and
(orientation : landscape) {
    /* 样式设计 */
}
/* iPads (横屏) ----- */
@media only screen and (min-device-width : 768px) and (max-device-width : 1024px) and
(orientation : portrait) {
    /* 样式设计 */
}
/* 台式机和笔记本 ----- */
@media only screen and (min-width : 1224px) {
    /* 样式设计 */
}
/* 超大屏 ----- */
@media only screen and (min-width : 1824px) {
    /* 样式设计 */
}
/*苹果 iPhone 4 ----- */
@media only screen and (-webkit-min-device-pixel-ratio:1.5), only screen and (min-device-
pixel-ratio:1.5) {
```

```
/* 样式设计 */
}
```

(5) CSS 字体属性的简写（或缩写）

```
body {
    font: font-style font-variant font-weight font-size line-height font-family;
}
```

(6) 自定义文本选择的高亮效果

```
::selection { background: #e2eae2; }
::-moz-selection { background: #e2eae2; }
::-webkit-selection { background: #e2eae2; }
```

(7) 锚链接伪类的设置

```
a:link { color: blue; }
a:visited { color: purple; }
a:hover { color: red; }
a:active { color: yellow; }
```

(8) 全屏背景

```
html {
    background: url('images/bg.jpg') no-repeat center center fixed;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
}
```

(9) 垂直居中内容

```
.container {
    min-height: 6.5em;
    display: table-cell;
    vertical-align: middle;
}
```

(10) 强制垂直滚动条

```
html { height: 101% }
```

(11) 设置自定义字体

```
@font-face {
    font-family: 'MyWebFont';
    src: url('webfont.eot'); /* IE9 Compat Modes */
    src: url('webfont.eot?#iefix') format('embedded-opentype'), /* IE6-IE8 */
    url('webfont.woff') format('woff'), /* Modern Browsers */
    url('webfont.ttf') format('truetype'), /* Safari, Android, iOS */
    url('webfont.svg#svgFontName') format('svg'); /* Legacy iOS */
}
body {
    font-family: 'MyWebFont', Arial, sans-serif;
}
```

（12）自定义段落首字母

```
p:first-letter{
    display: block;
    margin: 5px 0 0 5px;
    float: left;
    color: #ff3366;
    font-size: 5.4em;
    font-family: Georgia, Times New Roman, serif;
}
```

（13）CSS 3 盒子模型内部阴影

```
#mydiv {
    -moz-box-shadow: inset 2px 0 4px #000;
    -webkit-box-shadow: inset 2px 0 4px #000;
    box-shadow: inset 2px 0 4px #000;
}
```

（14）CSS 3 盒子模型外部阴影

```
#mydiv {
    -webkit-box-shadow: 0 2px 2px -2px rgba(0, 0, 0, 0.52);
    -moz-box-shadow: 0 2px 2px -2px rgba(0, 0, 0, 0.52);
    box-shadow: 0 2px 2px -2px rgba(0, 0, 0, 0.52);
}
```

（15）固定宽度的居中布局

```
#page-wrap {
    width: 800px;
    margin: 0 auto;
}
```

（16）禁用移动 Webkit 浏览器中的高亮显示

```
body {
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}
```

（17）CSS 省略号

```
div{
    width:200px;                /*设置宽度*/
    white-space:nowrap;         /*设置不折行*/
    text-overflow:ellipsis;     /*这就是省略号喽*/
    -o-text-overflow:ellipsis;  /*兼容 Opera*/
    overflow: hidden;           /*将超出的部分设置为隐藏*/
}
```


(18) CSS 3 斑马条纹

```
tbody tr:nth-child(odd) {
    background-color: #ccc;
}
```

(19) 给浏览器的滚动条加上颜色

```
body{
    scrollbar-face-color:#666666;
    scrollbar-shadow-color:#FFFFFF;
    scrollbar-highlight-color:#FFFFFF;
    scrollbar-3dlight-color:#3366cc;
    scrollbar-darkshadow-color:#666666;
    scrollbar-track-color:#EEEEEE;
    scrollbar-arrow-color:#666666
}
```

(20) 文本动画

```
ul {
    animation:6s linear 0 normal none infinite change;
    -webkit-animation:6s linear 0 normal none infinite change;
    -moz-animation:6s linear 0 normal none infinite change;
    -o-animation:6s linear 0 normal none infinite change;
}

@-webkit-keyframes change {
    0%    {margin-top:0;}
    15%   {margin-top:0;}
    25%   {margin-top:-40px;}
    40%   {margin-top:-40px;}
    50%   {margin-top:-80px;}
    65%   {margin-top:-80px;}
    75%   {margin-top:-40px;}
    85%   {margin-top:-40px;}
    100%  {margin-top:0;}
}

@keyframes change {
    0%    {margin-top:0;}
    15%   {margin-top:0;}
    25%   {margin-top:-40px;}
    40%   {margin-top:-40px;}
    50%   {margin-top:-80px;}
    65%   {margin-top:-80px;}
    75%   {margin-top:-40px;}
    85%   {margin-top:-40px;}
    100%  {margin-top:0;}
}
```

(21) 文字阴影

```
text-shadow: 5px 5px 5px #6600ff;
```

(22) 文字的水平居中

```
.container{  
    text-align:center;  
}
```

(23) 文字的垂直居中

```
.container {  
    height: 35px;  
    line-height: 35px;  
}
```

(24) 让子元素相对于父元素垂直居中

```
<div id="big">  
    <div id="small">  
    </div>  
</div>  
#big {  
    position: relative;  
    height: 480px;  
}  
  
#small {  
    position: absolute;  
    top: 50%;  
    height: 240px;  
    margin-top: -120px;  
}
```

(25) 图片宽度的自适应

```
img {  
    max-width: 100%  
}  
_img {  
    width: 100%  
}
```

(26) 简单 3D 按钮

```
#button {  
    background: #888;  
    border: 1px solid;  
    border-color: #999 #777 #777 #999;  
}
```

(27) 用图片美化列表标志

```
ul {  
    list-style: none  
}
```

```
ul li {
  background-image: url("path-to-your-image");
  background-repeat: none;
  background-position: 0 0.5em;
}
```

(28) 用 CSS 画三角形

```
.triangle {
  border-color: transparent transparent green transparent;
  border-style: solid;
  border-width: 0px 300px 300px 300px;
  height: 0px;
  width: 0px;
}
```

(29) 禁止自动换行

```
h1 {
  white-space: nowrap;
}
```

(30) 使用图片替换<h1>标签的元素又不影响 SEO

```
h1 {
  text-indent: -9999px;
  background: url("h1-image.jpg") no-repeat;
  width: 200px;
  height: 50px;
}
```

本书浏览器约定

本书涉及的浏览器测试基准如图 1 所示，可以看到内核和外壳的占有情况，内环为内核，外环为外壳，这也是接下来要提到的浏览器同类项的数据基础。

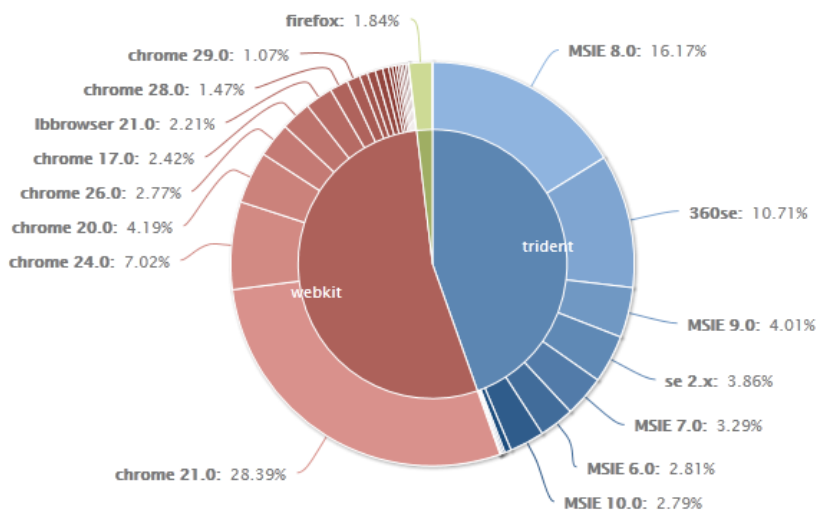


图 1 浏览器基准

不同的浏览器可能会采用相同的内核（渲染引擎）。一方面，外壳实现的差异性会影响整个网站功能，这也是为什么内核无法完全代表所有浏览器的原因。另一方面，某些浏览器具有相同的血统，是可以归为同一类的，同一类中，如果测试通过其中一个，那么其他与之同类的浏览器也基本上可以通过测试。图 1 是根据相同的内核或对 W3C 标准有类似的支持程度将浏览器进行归类的。之所以这样做，是因为测试基准除了要基本反映浏览器的市场占有率，还要考虑到开发测试成本。因此，测试基准中的浏览器应当具有典型代表性。

浏览器的不同版本的内核也不一样，因此通常要针对不同版本的浏览器作测试，开发者要了解内核对标准的支持，比如 IE 的渲染引擎 Trident 的不同版本差别较大，因此 IE 需要测试 6~10 的所有版本，然而由于 Firefox 和 Chrome 升级覆盖面广，因此基准中只保留其最新版本。

本书涉及的代码会根据需要，对不同浏览器进行针对性的测试。

序 2 你绝对不可能全部 做对的 CSS 题



全世界所有做过这些题目的 CSSer 的正确率是 58%，在中国相当于不及格！不用百度搜索答案，试试你是否能及格？如果不用百度，你就全做对了，相信本书不是你的菜！如果你的正确率不到 58%，每天坚持看本书，会带来意想不到的技术飞跃！

1. CSS 对大小写敏感（CSS 区分大小写）吗？
A、是
B、否
2. 对行内元素设置 `margin-top` 或 `margin-bottom` 属性是否有效？
A、是
B、否
3. 对行内元素设置 `padding-top` 或 `padding-bottom` 属性是否有效？
A、是
B、否
4. 假设对 `<p>` 元素设置样式 `font-size:10rem`，当用户改变浏览器窗口大小时，该元素内字体大小是否会随着窗口大小变化而发生变化？
A、是
B、否
5. 伪类 `:checked` 仅适用于 `type` 属性为 `radio` 或 `checkbox` 的 `input` 表单，并不适用于 `<option>` 表单，该说法正确吗？
A、正确
B、错误
6. 在 HTML 文档中，伪类 `:root` 永远是 `<html>` 元素，该说法正确吗？
A、正确
B、错误
7. `translate()` 方法可以在 `z` 轴方向上移动元素的位置，该说法正确吗？
A、正确
B、错误

8. 有如下代码：

```
//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
ul {
  color: red;
}
li {
  color: blue;
}
```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

9. 有如下代码：

```
//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
ul li {
  color: red;
}
#must-buy {
  color: blue;
}
```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

10. 有如下代码：

```
//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
.shopping-list .favorite {
  color: red;
}
```

```

}
#must-buy {
  color: blue;
}

```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

11. 有如下代码：

```

//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
ul#awesome {
  color: red;
}
ul.shopping-list li.favorite span {
  color: blue;
}

```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

12. 有如下代码：

```

//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
ul#awesome #must-buy {
  color: red;
}
.favorite span {
  color: blue!important;
}

```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

13. 有如下代码：

```
//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
ul.shopping-list li .highlight {
  color: red;
}
ul.shopping-list li .highlight:nth-of-type(odd) {
  color: blue;
}
```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

14. 有如下代码：

```
//HTML 代码
<ul class="shopping-list" id="awesome">
  <li><span>Milk</span></li>
  <li class="favorite" id="must-buy"><span class="highlight">Sausage</span></li>
</ul>
//CSS 代码
#awesome .favorite:not(#awesome) .highlight {
  color: red;
}
#awesome .highlight:nth-of-type(1):nth-last-of-type(1) {
  color: blue;
}
```

文本 Sausage 将呈现什么颜色？

- A、红色
- B、蓝色
- C、以上都不是

15. 有如下代码：

```
//HTML 代码
<p id="example">Hello</p>
//CSS 代码
#example {
  margin-bottom: -5px;
}
```

元素#example 的位置将会发生什么变化？

- A、下移 5px
- B、#example 所有的后继节点将上移 5px
- C、以上都不是

16. 有如下代码:

```
//HTML 代码
<p id="example">Hello</p>
//CSS 代码
#example {
  margin-left: -5px;
}
```

元素#example 的位置将会发生什么变化?

- A、左移 5px
- B、#example 所有父元素将右移 5px
- C、以上都不是

17. 有如下代码:

```
//HTML 代码
<div id="test1">
  <span id="test2"></span>
</div>
//CSS 代码
#i-am-useless {
  background-image: url('mypic.jpg');
}
```

未使用的样式表里的资源文件是否会被浏览器下载?

- A、是
- B、否

18. 有如下代码:

```
//HTML 代码
<div id="test1">
  <span id="test2"></span>
</div>
//CSS 代码
#test2 {
  background-image: url('mypic.jpg');
  display: none;
}
```

页面加载时, 图片 mypic.jpg 是否会被浏览器下载?

- A、是
- B、否

19. 有如下代码:

```
//HTML 代码
<div id="test1">
```

```
<span id="test2"></span>
</div>
//CSS 代码
#test1 {
    display: none;
}
#test2 {
    background-image: url('mypic.jpg');
    visibility: hidden;
}
```

页面加载时，图片 mypic.jpg 是否会被浏览器下载？

- A、是
- B、否

20. 有如下代码：

```
//CSS 代码
@media only screen and (max-width: 1024px) {
    margin: 0;
}
```

选择器 only 的作用是：

- A、阻止老版本浏览器解析剩下的选择器
- B、仅对屏幕设备应用样式，忽略设备的 max-width
- C、什么都不做

21. 有如下代码：

```
//HTML 代码
<div>
    <p>I am floated</p>
    <p>So am I</p>
</div>
//CSS 代码
div {
    overflow: hidden;
}
p {
    float: left;
}
```

样式 overflow: hidden 是否会创建一个新的块格式上下文？

- A、是
- B、否

22. 有如下代码：

```
//CSS 代码
@media only screen and (max-width: 1024px) {
    margin: 0;
}
```

关键字 `screen` 是适用于设备的物理屏幕还是浏览器的窗口：

- A、设备物理屏幕
- B、浏览器窗口

23. `<keygen>` 是否是合法的 HTML 5 标签？

- A、是
- B、否

24. `<bdo>` 标签是否改变文本方向？

- A、是
- B、否

25. 有如下代码：

```
//HTML 代码
<figure>
  
  <figcaption>
    <p>This is my self portrait.</p>
  </figcaption>
</figure>
```

以上 HTML 代码是否合法：

- A、是
- B、否

26. 在以下什么情况下可以使用 `<small>` 标签：

- A、当需要在 `<h1>` 标签之后创建子标题时
- B、当需要在 `<footer>` 标签内添加版权信息时

27. 一个网页上含有结构性完整的多个 `<h1>` 标签，是否会优化 SEO：

- A、是
- B、否

28. 一个搜索结果页包含多条搜索结果，用哪个 HTML 标签突出标记搜索词：

- A、``
- B、`<mark>`
- C、``
- D、`<highlight>`

29. 有如下代码：

```
//HTML 代码
<article>
  <h1>Hello World</h1>
  <style scoped>
    p {
      color: #FF0;
    }
  </style>
```

```
<p>This is my text</p>
</article>

<article>
  <h1>This is awesome</h1>
  <p>I am some other text</p>
</article>
```

属性 `scoped` 有什么作用？

- A、将样式规则应用于与元素居右相同父节点的后继节点上
- B、将样式规则应用于 `scoped` 标记的父元素的所有子节点
- C、仅将样式规则应用于 IE 浏览器

30. 有如下代码：

```
//HTML 代码
<article>
  <a href="#">
    <h1>Hello</h1>
    <p>I am some text</p>
  </a>
</article>
```

HTML 5 是否支持块级链接？

- A、是
- B、否

31. 有如下代码：

```
//HTML 代码

```

以上 HTML 代码是否会触发 HTTP 请求？

- A、是
- B、否

32. 有如下代码：

```
//HTML 代码
<div style="display: none;">
  
</div>
```

以上 HTML 代码是否会触发 HTTP 请求？

- A、是
- B、否

33. 有如下代码：

```
//HTML 代码
<head>
  <link href="main1.css" rel="stylesheet">
  <script>
    alert('Hello World');
```

```
</script>
</head>
```

在 Hello World 被 alert 出来之前，main1.css 文件是否被浏览器下载并解析？

- A、是
- B、否

34. 有如下代码：

```
//HTML 代码
<head>
  <link href="main1.css" rel="stylesheet">
  <link href="main2.css" rel="stylesheet">
</head>
```

在 main2.css 文件被获取之前，main1.css 文件必须先被浏览器下载并解析？

- A、是
- B、否

35. 有如下代码：

```
//HTML 代码
<head>
  <link href="main1.css" rel="stylesheet">
</head>
<body>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
  <link href="main2.css" rel="stylesheet">
</body>
```

在 Paragraph 1 被渲染之前，main1.css 文件是否被浏览器下载并解析？

- A、是
- B、否

是不是已经迫不及待想知道这些题目的答案了？

- (1) 添加微信公众号“broadview_com”，或扫描封底的微信二维码。回复关键字“CSS 题答案”，核查你的正确率吧！
- (2) 分享你的正确率，邀请小伙伴们都来做题，看看谁才是隐藏在民间的高手！
- (3) 正确率超过 80%的小伙伴们可以@博文视点 Broadview#，我们会与全球读者一起分享你的得分超能力！



序 3 最流行的前端面试题

面试题要考的不是多么高精尖的问题，你肯定想不到，所有世界 500 强和中国 100 强的互联网科技公司，所有的考题都是基础技术题，因为他们招聘的不是经理，是干活的，所以要考查的是你的基本功！主要考量对 Web 标准的理解、浏览器兼容差异、CSS 基本功，如布局、盒子模型、选择器优先级及使用、HTML 5、CSS 3、移动端开发技术等基础。面试题同样不做解释，读者可自己测试！

1. Doctype 作用是什么？严格模式与混杂模式分别是如何触发这两种模式的，区分它们有何意义？
2. 行内元素有哪些？块级元素有哪些？空(void)元素有哪些？
3. 如何理解 CSS 的盒子模型？
4. link 和@import 的区别是什么？
5. CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？CSS3 新增伪类有哪些？
6. 如何居中 div，如何居中一个浮动元素？
7. 浏览器的内核分别是什么？经常遇到的浏览器的兼容性有哪些？原因、解决方法是什么，常用 Hack 的技巧有哪些？
8. HTML5\CSS3 有哪些新特性、移除了哪些元素？如何处理 HTML5 新标签的浏览器兼容问题？
9. 你怎么来实现页面设计图，你认为前端应该如何高质量完成工作？
10. 列出 display 的值，说明它们的作用。position 的值里，relative 和 absolute 定位原点是什么？
11. 页面重构怎么操作？
12. 语义化的理解？
13. HTML5 的离线储存？
14. 为什么要初始化 CSS 样式？
15. 对 BFC 规范的理解？
16. iframe 有哪些缺点？
17. CSS 是怎样定义权重规则的？
18. 如何理解表现与内容相分离？
19. 如何解决 IE6 的双边距问题？

20. 如何定义高度为 1px 的容器?
21. 如何解决 IE 6 的 3 像素问题?
22. Firefox 下文本无法撑开容器的高度, 如何解决?
23. 怎么样才能让层显示在 Flash 之上呢?
24. `cursor:hand` 在 FF 不显示小手, 如何解决?
25. 在 IE 中内容会自适应高度, 而 FF 不会自适应高度, 怎么办?
26. 前端页面有哪三层构成, 分别是什么? 作用是什么?
27. 你做的页面在哪些浏览器测试过? 这些浏览器的内核分别是什么? 经常遇到的浏览器的兼容性有哪些? 怎么会出现? 解决方法是什么?

这里没有标准答案, 但你可能已经有了自己的答案! 添加微信公众号“broadview_com”, 回复关键字“CSS 面试题答案”, 看看笔者给的参考答案和你自己的答案有什么不同。

目 录

序 1 CSS 的前世今生	VII
---------------------	-----

序 2 你绝对不可能全部做对的 CSS 题	XVII
--------------------------------	------

序 3 最流行的前端面试题	XXVI
---------------------	------

第 1 章 文字与字体	1
-------------------	---

1.1 在网页中使用自定义字体	1
1.2 文本缩进和首字符下沉	3
1.3 自定义文本被选中时的样式	4
1.4 文本对齐	4
1.5 调整文字、字符的间距	5
1.6 文本的装饰——画线、粗体、斜体	6
1.7 文字阴影	6
1.8 文字毛玻璃效果	7
1.9 文本溢出处理	7
1.10 金属质感文字	8
1.11 隐藏文本	9
1.11.1 使用 text-indent	9
1.11.2 使用定位	9
1.12 文字旋转	10
1.13 现代字体栈	10

第 2 章 边框和图片	12
-------------------	----

2.1 边框新属性的基础与实例	12
2.1.1 border-color	12
2.1.2 border-image	13
2.1.3 border-radius	17
2.1.4 box-shadow	18
2.2 搜索框	19
2.2.1 使用背景图片的搜索框	19
2.2.2 只使用 CSS 的搜索框	20
2.3 微博发布框	21

2.4 拍立得效果框	26
2.5 CSS 3 动画边框	27
2.6 边框移动特效	31
2.7 Banner 图片的标签	32
2.8 黑白图片	34
2.9 图片水印	34
2.10 图片细节放大展示	35
2.11 图片的瀑布流	39
2.11.1 浮动的瀑布流	40
2.11.2 绝对定位的瀑布流	42
2.12 图片墙	45
2.13 图片轮播图	47
2.13.1 使用定位实现	48
2.13.2 使用透明度实现	52
2.13.3 无缝切换	52
2.14 幻灯片	56
2.15 手风琴效果	60
2.16 图片自适应	61
2.17 使用纯 CSS 绘制图像	62
2.18 图片原地放大	66
2.19 图片翻转	67
2.20 图像地图	68

第 3 章 按钮和链接	71
-------------------	----

3.1 圆角按钮	71
3.2 简单导航栏	74
3.3 二级导航栏	76
3.4 三级导航栏	78
3.5 滑动菜单	80
3.6 网页右键菜单	82
3.7 下拉菜单	84
3.8 CSS 3 圆形导航菜单	87
3.9 标签云	90
3.10 TAB 标签页	91

3.10.1 使用 JavaScript	92	5.6 进度条	139
3.10.2 使用 CSS target 伪类	92	5.7 图标滑动切换特效	142
3.11 选中文字分享	93	5.8 流星划过效果	143
3.12 链接百叶窗效果	94	5.9 雪花飘落效果	144
3.13 iPhone 开关	96	5.10 数字滚动器	146
3.14 按钮式单选框与复选框	97	5.11 模拟时钟	149
3.15 自定义播放器	100	5.12 苹果著名的 DOCK 栏	154
3.16 文字变链接	104	5.13 苹果系统的 Stack 特效	158
3.17 根据文件格式设置链接图标	104	5.14 扇形展开	161
3.18 链接标签“a”的顺序	105	5.15 回到页面的顶部	166
第 4 章 背景和颜色	108	5.16 拖曳和抛出	167
4.1 颜色和渐变的基础与实例	108	5.16.1 拖曳实现原理	167
4.1.1 颜色	108	5.16.2 抛出与模拟抛物原理	168
4.1.2 渐变简述	109	5.16.3 窗口实现	168
4.1.3 带前缀的渐变	109	第 6 章 页面的布局	170
4.1.4 W3C 标准渐变（不带前缀）	111	6.1 图文混排	170
4.1.5 重复渐变	111	6.2 文本内容垂直居中	172
4.2 高光效果	112	6.3 自适应宽度的水平居中	173
4.3 多背景	112	6.4 固定宽度且居中	174
4.4 全屏背景	113	6.5 固定页脚	175
4.5 斑马线背景	114	6.6 控制位置：绝对位置和相对位置	177
4.6 棋盘背景	115	6.7 一个图文混排的网页选项卡	178
4.7 易拉罐效果	117	6.8 兼容浏览器的最小高度	182
4.8 页面顶部阴影	119	6.9 让 div 显示在屏幕的中央	183
第 5 章 变换与动画	120	6.10 iPad 屏幕布局	185
5.1 CSS 3 变换与动画的基础及实例	120	6.11 经典的 CSS Clearfix	186
5.1.1 CSS 3 变形概述	120	6.12 升级版的 Clearfix	187
5.1.2 CSS 3 变形语法详解及应用	121	6.13 强制垂直滚动条	189
5.1.3 CSS 3 转换概述	122	6.14 CSS 3 文本分列	190
5.1.4 CSS 3 转换语法详解	123	6.15 让 div 层在 Flash 之上	192
5.1.5 CSS 3 转换具体实例	124	6.16 float 引起 div 自适应高度无效的解 决方案	193
5.1.6 CSS 3 动画概述	124	6.17 Flexbox 布局风格	196
5.1.7 CSS 3 动画语法详解	125	6.18 动态高度下的居中	201
5.1.8 简单实例	127	6.19 纯 CSS 实现固定表头	202
5.2 纸张边角动画效果	127	6.20 Metro 布局风格	205
5.2.1 纸张边角稍稍卷起	128	第 7 章 美化与装饰	209
5.2.2 边角翻折	130	7.1 文本装饰	209
5.2.3 更具立体感的边角翻折效果	132	7.1.1 文本的颜色	209
5.3 气泡式提示	134	7.1.2 文本画线	210
5.4 对联广告	136	7.1.3 文本的空白	212
5.5 页面 loading 效果	137		

7.1.4 文本的方向.....	212	10.6 CSS 3 滤镜.....	286
7.2 发光输入框.....	213	10.7 常用的 CSS Hack 列表.....	291
7.3 自定义滚动条.....	214	10.8 CSS 重置方案 (CSS Reset).....	294
7.4 页面顶部阴影.....	218	10.8.1 方案一.....	294
7.5 巧妙实现分隔线.....	218	10.8.2 方案二 (雅虎方案).....	295
7.6 三角形列表符号.....	221	10.8.3 方案三.....	295
7.7 纸页面卷曲效果.....	222		
7.8 跨浏览器的透明度.....	225	第 11 章 其他常用代码.....	297
7.9 鼠标指向时变成手型.....	227	11.1 使用 CSS 3 实现简单的计算器.....	297
7.10 鼠标移动到 div 上高亮显示.....	227	11.2 使用 CSS 3 制作网页播放器.....	301
7.11 发光锚链接.....	229	11.3 不使用 table 的 Form 表单.....	306
7.12 屏蔽 Webkit 浏览器的高亮效果.....	230	11.4 可以重复利用的规则.....	310
7.13 多种风格的翻页页码.....	232	11.5 在同一元素上使用多种类.....	312
7.13.1 Yahoo 旧版翻页风格.....	232	11.6 CSS 块引用模板.....	313
7.13.2 Yahoo 新版翻页风格.....	234	11.7 花式 CSS 3 Pull-引文.....	314
7.13.3 Meneame 翻页风格.....	236	11.8 一般媒体查询.....	315
7.13.4 YouTube 翻页风格.....	238	11.9 CSS 3 背景梯度.....	317
7.14 创建针线缝合效果.....	239	11.10 CSS 日历显示效果.....	318
第 8 章 盒子.....	241	11.11 字符编码.....	322
8.1 CSS 3 盒模型.....	241	11.12 手机 APP 使用的简洁注册页面.....	323
8.2 内层 CSS 3 盒阴影.....	243	11.13 手机简洁价目表.....	327
8.3 外层 CSS 3 盒阴影.....	243	11.14 手机简洁任务表.....	331
8.4 纯 CSS 3 透明水晶盒.....	244	11.15 微店购物车.....	334
8.5 投影发光效果.....	249	11.16 APP 导航与提醒.....	338
第 9 章 3D 相关.....	251	11.17 简洁记事本.....	341
9.1 3D 文字.....	251	11.18 手机文件下载.....	345
9.2 3D 图片立体效果.....	253	11.19 迷你下拉列表框.....	347
9.3 3D 按钮.....	258	11.20 Google Font API.....	349
9.4 3D 下拉菜单.....	264	11.21 动态提示框.....	350
9.5 3D 旋转动画.....	269	11.22 用 CSS 创建内容幻灯片.....	353
第 10 章 CSS Hack.....	273	11.23 打印自动显示超链接 URL.....	357
10.1 让网站在所有浏览器下显示一致 (CSS Reset).....	273	11.24 禁用 Webkit 内核某些属性.....	357
10.2 解决 IE 6 中浮动元素的双倍边距 问题.....	276	11.24.1 禁用电话号码转换为 链接样式 (移动设备).....	357
10.3 识别不同浏览器.....	277	11.24.2 禁用原生弹出菜单 (移动设备).....	357
10.4 背景与图片透明.....	281	11.24.3 禁用用户选中.....	358
10.5 IE 10 CSS Hack.....	285	11.24.4 禁用输入框、文本框的 轮廓线.....	358
		11.24.5 禁用文本框的缩放功能.....	358

第 1 章 文字与字体



文字是网页中最基本的元素，因为网络上大部分用户都是以查找信息为目的，所以文字几乎是多数网站的灵魂。一般说来，网站访客不会很在意网站文本的样式，但是文字的设置是非常必要的，多种样式的文字能够加强网站内容的层次性，能够突出重点，使网页内容清晰合理，奇特的文本样式也会给人耳目一新的感受。从本质上讲，CSS 对文字的设置与 Word 中格式化文字是一样的。本章将介绍 CSS 在设置文字与字体方面的某些特性及应用实例。

本章主要涉及的知识点有：

- 自定义字体、自定义文本被选中的样式
- 文本缩进与首字符下沉
- 文本对齐、文字间距、文本溢出
- 文字阴影、毛玻璃效果、金属质感等装饰
- 隐藏文本、文字旋转
- 现代字体栈

有关字体的设置都较为简单，因此，本章的实例是一些较为基础的应用。

1.1 在网页中使用自定义字体

CSS 3 中新增@font-face 规则，该规则允许网页设计人员在网页中使用自己喜欢的字体，这些字体在访客的计算机中可能不存在，但是当网页设计人员把这些字体放置在服务器之后，浏览器会下载这些字体集，然后展示给访客。

在 CSS 2 中，font-family 属性只能使用两种字体：

(1) 一种是通用字体系列（拥有相似外观的字体系统组合，包括 Serif 字体、Sans-serif 字体、Monospace 字体、Cursive 字体和 Fantasy 字体）；

(2) 一种是特定字体系列（具体的字体系列，如 Times 或 Courier），它只能使用用户计算机中已经安装的字体，这也是它的局限性。如果用户计算机里没有安装需要的字体，浏览器会使用 CSS 中定义的第一个用户已有的字体类型，如果最终没有 CSS 所需的字体，那么浏览器将使用默认的字体类型。

相对于 CSS 2，CSS 3 这一新规则使得网页更加多样性。这一规则的应用多见于很多国外的网站。图 1.1 为 IE 和谷歌浏览器下使用徐静蕾字体的网页效果。虽说不同浏览器下的

效果相同，但实际上不同浏览器对字体文件格式的支持是不同的。微软的 IE 5 最先支持这个属性，但是只支持微软自有的.eot (Embedded Open Type) 格式，除 IE 之外的浏览器直到现在都没有支持这一字体格式。从 Safari 3.1 开始，支持.ttf (TrueType) 和.otf (OpenType) 两种字体，截止到现在，IE 之外的主流浏览器都已支持这些格式。

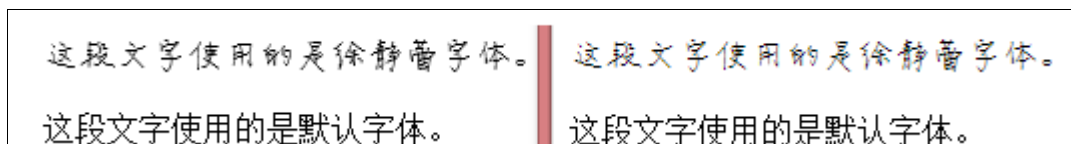


图 1.1 IE 浏览器和谷歌 Chrome 浏览器下使用徐静蕾字体的网页效果

上述效果实现的代码如下：

```
01  @font-face
02  {
03      font-family: xujinglei;           /*定义字体的名称*/
04      src: url('xujinglei.ttf')        /*字体在服务器的位置*/
05      ,url('xujinglei.eot');           /* IE9+ */
06  }
07  div
08  {
09      font-family:xujinglei;           /*使用已经定义的字体*/
10      font-size:18px;
11  }
```

在使用自定义字体之前，首先在第 3 行定义字体的名称，第 4、5 行指明字体的相对路径。为了兼容，一般定义两种格式的字体即可。第 9 行在使用时与 CSS 2 无异。

自定义字体在英文网站很常见，但是目前在中文网站上却很少。原因有二：

(1) 英文只有 26 个字母以及其他字符，字体文件占用空间小，网站使用自定义字体之后对网站的响应速度并无明显影响。

(2) 国外的网速快，即使是大文件，也会很快的下载完成，用户不会感觉网站响应慢的问题。而相比之下，中文的字体文件较大，加上目前国内的网速还处在缓慢的状态，使用自定义字体虽然美观，却一定会拖慢网站的运行，所以目前在中文网站上很少存在大面积自定义字体的现象。但是，随着中国互联网的发展，光纤和 4G 技术的进步，相信在不久的将来一定会实现飞速的网络，届时，自定义的字体在中文网站的普及也就没有压力了，@font-face 终将有有用武之地。

注意：在只能找到.ttf 格式的字体文件时，建议采用 EOTFast 软件将 ttf 格式转化为 eot 格式，在 ttf 格式文件上点击右键，选择“打开方式”快捷菜单，用 EOTFast (<http://www.eotfast.com/>) 打开，即可自动生成同名 eot 文件。另外在代码文件夹中也提供了 EOTFast 软件。有时网站也会使用 svg 格式的字体，图片字体经常被应用在使用字体中的图标以取代图片。

1.2 文本缩进和首字符下沉

通常在报刊和杂志上，我们都会看到段首字符下沉效果，这在信息类网页中也会经常用到，而文本缩进则是段落排版不可或缺的风格。首字符下沉和文本缩进效果如图 1.2 所示。

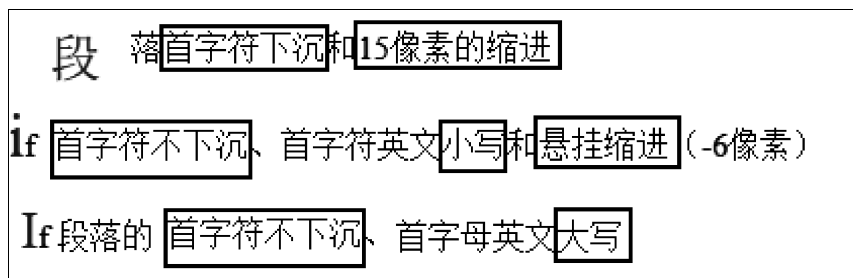


图 1.2 首字符下沉和文本缩进效果

首字符下沉效果用 CSS 2 中的 :first-letter 伪类实现：

```

01  #a{
02      text-indent:15px;                /*使用像素作为单位*/
03  }
04  #b{
05      text-indent:-6px;
06  }
07  #c{
08      text-indent:0.5em;                /*使用 em 作为单位*/
09  }
10  #a:first-letter{
11      font-size:25px;
12      color:#B23AEE;
13      float:left;                        /*要下沉必须浮动*/
14  }
15  #b:first-letter{
16      font-size:30px;
17      color:red;
18  }
19  #c:first-letter{
20      font-size:25px;
21      color:#B23AEE;
22  }
23  p.uppercase:first-letter
24  {
25      text-transform:uppercase;          /*大写的关键字*/
26  }

```

第 2 行、第 5 行和第 8 行是对不同文字设置不同的缩进，由图中 3 段文字的缩进情况可以对比结果。利用 `first-letter` 匹配的每段的第一个字母，段落之间不同之处只有第 13 行的浮动，一旦首字母浮动在左边，就会自己得到浮动的效果，图中第一段文字显示首字母在整排文字的下方线上。第 25 行利用大写关键字可以设置字母的大写显示，可以将 HTML 文档中的小写英文字母在不改动的情况下变为大写字母，对中文无效。

1.3 自定义文本被选中时的样式

当鼠标选中网页上的文本时，默认的样式为蓝底白字，在 CSS 3 中，这种样式是可以自定义的。CSS 3 中有 UI 元素状态伪类 `::selection`，该属性可以设置文本选中的效果。图 1.3 为默认样式与自定义样式的对比图。



图 1.3 文本选中效果对比

CSS 代码较为简单，使用 `E::selection: {CSS 样式}` 即可，图中效果的代码如下：

```
01  ::selection
02  {
03      color:#9400D3;           /*字体颜色*/
04      background:#A9A9A9;      /*背景颜色*/
05  }
06  ::-moz-selection             /*Firefox 下的设置*/
07  {
08      color:#9400D3;
09      background:#A9A9A9;
10  }
```

注意：在老版本的 Firefox 下，可能需要加 `-moz-` 前缀做兼容。与其他伪类不同，这一伪类的前面有两个冒号，请注意不要遗漏。

1.4 文本对齐

文本对齐方式也是排版中常用的属性，由于不同语言的阅读顺序可能不同，所以网页设计时要考虑文本对齐。文本对齐使用属性 `text-align`，它有 3 个值可以选择：

```
01  text-align: center          /*文本居中对齐*/
02  text-align: left             /*文本左对齐*/
03  text-align: right            /*文本右对齐*/
```

文本居中经常用在将文字调整到元素的中央位置上，如要文字位于按钮的中央，通常

使用下面的代码：

```
01 text-align:center;
02 line-height:15px;
```

其中第 1 行使得按钮上的文字水平居中，第 2 行中的 `line-height` 为行高的设置，如同 Word 中对文字行高的设置一样，当指定一行文字的高度后，文字会在该行的中部显示（行高是影响上一行和下一行与本行距离的属性），当设置行高为按钮的高度时，按钮上方的文字自然显示在按钮的中部了。

注意：`text-align` 与 `center` 标签的效果似乎相同，实际上却是不相同的，因为 `center` 标签同时会使得元素居中。

1.5 调整文字、字符的间距

CSS 对间距的处理包含文字间距、行间距，以及对空格符、回车符和制表符的处理等多个方面。以下 CSS 代码包含常用的多种间距处理方式，具体效果见代码注释。

```
01 p.wordspacing{word-spacing:20px;}      /*设置空格的长度*/
02 p.letterspacing{letter-spacing:20px;}   /*设置字间距*/
03 p.lineheight{line-height:0.3;}         /*设置行间距*/
04 p.whitespace_normal{white-space:normal;} /*默认，忽略多个空格为 1 个，忽略回车符*/
05 p.whitespace_pre{white-space:pre;}      /*保留多个空格*/
06 p.whitespace_nowrap{white-space:nowrap;} /*忽略回车符，禁止换行，直到遇到 br*/
07 p.whitespace_prewrap{white-space:pre-wrap;} /*保留所有空格符与回车符*/
08 p.whitespace_preline{white-space:pre-line;} /*忽略多个空格为 1 个，保留回车符*/
```

以上代码执行之后的效果如图 1.4，中英文无异。

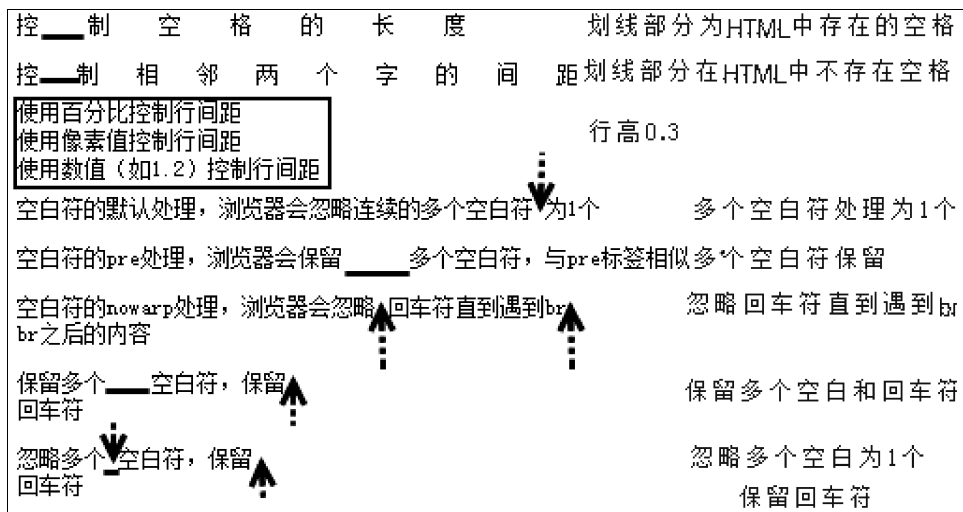


图 1.4 间距处理

注意：浏览器对制表符（Tab）的处理方式与空格符相同。

1.6 文本的装饰——画线、粗体、斜体

CSS 可以实现在字体上添加一条线的文本装饰效果，下面的 CSS 代码分别在文本的上方、文本的中央和文本的下方添加直线，设计粗体和斜体则更加基础。

```
01 .overline{text-decoration:overline;} /*上画线*/
02 .through{text-decoration:line-through;} /*穿过线*/
03 .underline{text-decoration:underline;} /*下画线*/
04 .blink{text-decoration:blink;} /*文本闪烁，暂不支持*/
05 .bold{font-weight:bold;} /*粗体*/
06 .italic{font-style:italic;} /*斜体*/
07 .oblique{font-style:oblique;} /*倾斜*/
```

不同样式的效果如图 1.5 所示。

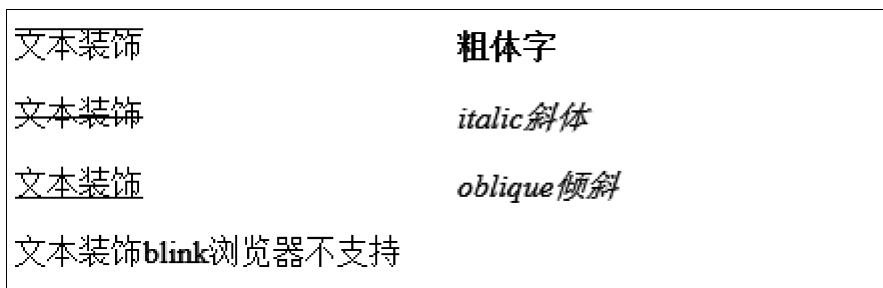


图 1.5 文本装饰

斜体 (italic) 是一种简单的字体风格，对每个字母的结构有一些小改动，来反映变化的外观，倾斜 (oblique) 文本则是正常竖直文本的一个倾斜版本，通常情况下，两种效果在浏览器中看上去完全一样。

在 a 链接标签上，经常把 text-decoration 的属性值取为 none，来取消链接的下画线。

注意：在 W3C 标准中，text-decoration 的值可以取 blink 得到文字闪烁的效果，不过 IE 浏览器、Chrome 浏览器和 Safari 浏览器目前不支持。

1.7 文字阴影

text-shadow 是 CSS 3 新增加的内容，利用它可以创造出文字阴影的效果，如图 1.6 所示。文字阴影与边框阴影同属于阴影范畴，所以 text-shadow 属性语法结构与 box-shadow 相似，参数均为水平偏移距离、竖直偏移距离、阴影宽度、阴影扩展和阴影颜色，前两个参数必选，边框阴影的应用参照本书边框和图片一章。在某些 3D 文字的场景中，文本阴影应用更为广泛。在同一文本上可以使用多个阴影，相邻两个之间用逗号隔开。

文字的阴影效果

图 1.6 文字阴影

图 1.6 的 CSS 代码是：

```
01 text-shadow: 5px 5px 5px #6600ff;
```

1.8 文字毛玻璃效果

使用文字阴影和字体透明颜色可以合成毛玻璃的效果，如图 1.7 所示。

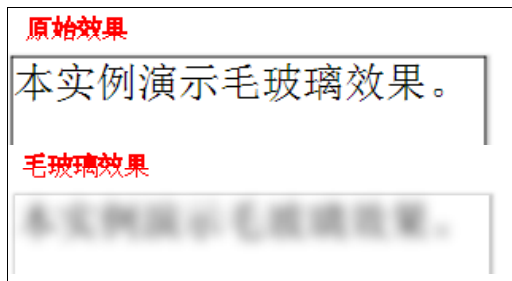


图 1.7 毛玻璃效果

其主要代码为：

```
01 box-shadow: 1px 1px 2px 2px #ccc;  
02 color: rgba(0,0,0,0);  
03 text-shadow: 0 0 10px black;
```

其中第 1 行为边框阴影，第 3 行为文字阴影，第 2 行通过透明度的设置使得文字变为不可见的状态，只留下文本阴影的模糊效果。

注意：使用完全透明的效果并不等同于使用白色，本实例若使用白色，可以清晰地看到白色字体，而不是模糊的效果。

1.9 文本溢出处理

网页设计中经常发生文本内容超出容器范围的问题，在内容较多时多采用分页效果，在内容较少时则采用一些普通的文本溢出处理。常用的文本溢出处理方式有：

- 简单裁切
- 简单隐藏
- 隐藏并显示省略号
- 使用滚动条

几种文本溢出的效果如图 1.8 所示。

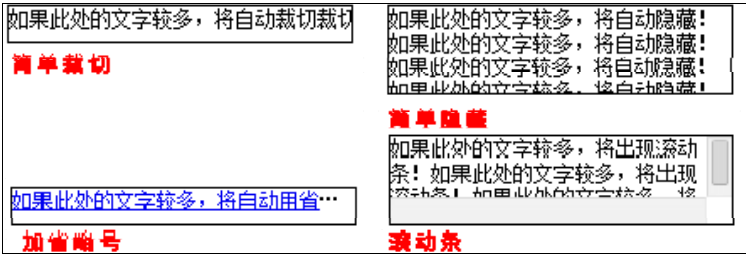


图 1.8 文本溢出处理

对应的 CSS 代码：

```
01  div.hide{overflow:hidden;}           /*简单隐藏*/
02  div.scroll{overflow:scroll;}         /*使用滚动条*/
03  div.clip{                           /*简单裁切*/
04      border:1px solid;
05      overflow:hidden;                 /*超出部分隐藏*/
06      white-space:nowrap;              /*强制在一行显示*/
07      text-overflow:clip; }            /*裁切*/
08  div.elli{                           /*超出部分省略号*/
09      overflow:hidden;                 /*超出部分隐藏*/
10      white-space:nowrap;              /*强制在一行显示*/
11      text-overflow:ellipsis; }        /*使用省略号*/
```

1.10 金属质感文字

文字的金属质感效果直接实现较为困难，本节效果的实现原理为在文本上方添加一个使用半透明和渐变效果的遮罩层，效果如图 1.9 所示。

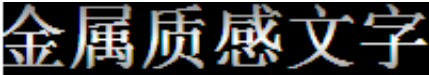


图 1.9 金属质感文字

CSS 代码如下：

```
01  p{
02      color:white;background:black;
03      font-weight:bold;font-size:30px;
04      position:relative;                /*对 p 定位才可以使浮动层绝对定位*/
05  }
06  span.cover{
07      width:100%;height:100%;position:absolute; /*浮动渐变层的定位*/
08      background:linear-gradient(to bottom,black 0%,transparent 50%,black);
```

```

09     opacity:0.5;                                /*设置透明度*/
10 }

```

在 `p` 标签中使用一个 `span`，该 `span` 的大小与父元素相同。为了实现金属质感，在第 2 行设置文字为黑底白字。`span` 上使用渐变，从顶部渐变到底部，从黑色渐变到透明再渐变到黑色（第 8 行渐变部分只使用了 W3C 标准，渐变的实现见其他章节）。在第 9 行对 `span` 的整体透明度进行了设置，这不是必要的，但可以使效果更好。

注意：`span` 部分也可以使用半透明的背景图片实现，只是不如渐变灵活。

1.11 隐藏文本

隐藏网页元素的方法有很多种，比如使用 `display:none`，或是使用全透明（`opacity`）。在对文本的处理中，有时并不希望文本丢失，而通常是把文字转移到屏幕的外面，换句话说，文字在屏幕之外，用户看不见，但它依然是存在的，浏览器可以检测到。本节介绍如下两种隐藏文本的方式：

- 使用 `text-indent` 隐藏——图片替换文本、搜索引擎优化。
- 使用定位隐藏——利于屏幕讲述工具的阅读。

1.11.1 使用 text-indent

`text-indent` 属性是之前提到的文本缩进，当文本缩进使用负参数缩进到屏幕之外时，文本就变得不可见了。此方法可以用图片代替文本，之前很多网站使用 `h1` 标签，把文本用此方法隐藏，在 `h1` 上设置 `logo` 为背景并设置居中来做搜索引擎优化（目前用此方法做搜索引擎优化效果已经过时）。

```

01 h1{
02     text-indent:-9999px;                          /*缩进*/
03     margin:0 auto;width:490px;height:200px;        /*居中*/
04     background:url("img/baidu.jpg") no-repeat;    /*背景图*/
05 }

```

1.11.2 使用定位

使用绝对定位使文本脱离文档流，然后把文本设置在屏幕的不可见区域即可。很多网站使用此法隐藏图标下的文字，但同时有利于“屏幕讲述者”阅读文本对视觉障碍者提供便利。其本质为绝对定位使得文字不可见。

```

01 .screen-reader-text {
02     position: absolute;                            /*绝对定位*/
03     top: -9999px;                                  /*顶部*/
04     left: -9999px;                                 /*左边*/
05 }

```

1.12 文字旋转

文字旋转可以使用 CSS 3 变形方便地实现，以下给出一段使文字从水平方向变为竖直方向的简单代码，对于文字旋转更多的实现，可以参照变换与动画一章中对 CSS 3 变形实现的介绍，在本节不做过多涉及，主要使用 `rotate` 旋转属性。

```
01 -webkit-transform:rotate(-90deg);      /*Webkit 内核浏览器*/
02 -moz-transform:rotate(-90deg);        /*火狐浏览器*/
03 -o-transform:rotate(-90deg);         /*旧版本 Opera 浏览器*/
04 -ms-transform:rotate(-90deg);        /*IE9+ 浏览器*/
05 transform:rotate(-90deg);            /*W3C 标准*/
```

1.13 现代字体栈

CSS 字体栈是一系列的字体，它包含了能在不同操作系统和平台上展示的字体，以尽可能地使排版保持一致性。浏览器会在 `font-family` 规定的所有字体中从前往后依次查找，如果找不到当前字体就查找下一个字体。网页设计的一个大目标是向大范围用户提供漂亮的排版，而且要考虑字体对所有的用户都有效，所以创建字体栈的原则是使用多个操作系统或终端所共有的或者交叉范围大的一系列字体，需要大量备用字体。

以下是几种常见的字体栈风格。

（1）基于 Times New Roman 的字体风格

```
font-family: Cambria, "Hoefler Text", Utopia, "Liberation Serif", "Nimbus Roman No9 L Regular", Times, "Times New Roman", serif;
```

（2）基于现代 Georgia 的字体风格

```
font-family: Constantia, "Lucida Bright", Lucidabright, "Lucida Serif", Lucida, "DejaVu Serif", "Bitstream Vera Serif", "Liberation Serif", Georgia, serif;
```

（3）基于 Garamond 的更传统的风格

```
font-family: "Palatino Linotype", Palatino, Palladio, "URW Palladio L", "Book Antiqua", Baskerville, "Bookman Old Style", "Bitstream Charter", "Nimbus Roman No9 L", Garamond, "Apple Garamond", "ITC Garamond Narrow", "New Century Schoolbook", "Century Schoolbook", "Century Schoolbook L", Georgia, serif;
```

（4）基于 Helvetica/Arial 的字体风格

```
font-family: Frutiger, "Frutiger Linotype", Univers, Calibri, "Gill Sans", "Gill Sans MT", "Myriad Pro", Myriad, "DejaVu Sans Condensed", "Liberation Sans", "Nimbus Sans L", Tahoma, Geneva, "Helvetica Neue", Helvetica, Arial, sans-serif;
```

（5）基于 Verdana 的字体风格

```
font-family: Corbel, "Lucida Grande", "Lucida Sans Unicode", "Lucida Sans", "DejaVu Sans", "Bitstream Vera Sans", "Liberation Sans", Verdana, "Verdana Ref", sans-serif;
```

（6）基于 Trebuchet 的字体风格

```
font-family: "Segoe UI", Candara, "Bitstream Vera Sans", "DejaVu Sans", "Bitstream Vera Sans",
```

```
"Trebuchet MS", Verdana, "Verdana Ref", sans-serif;
```

（7）更有深度的 Impac 字体风格

```
font-family: Impact, Haettenschweiler, "Franklin Gothic Bold", Charcoal, "Helvetica Inserat",  
"Bitstream Vera Sans Bold", "Arial Black", sans-serif;
```

（8）monospace 字体风格

```
font-family: Consolas, "Andale Mono WT", "Andale Mono", "Lucida Console", "Lucida Sans  
Typewriter", "DejaVu Sans Mono", "Bitstream Vera Sans Mono", "Liberation Mono", "Nimbus Mono  
L", Monaco, "Courier New", Courier, monospace;
```

注意：在很多博客或者 IDE（如 Dreamweaver）中有很多流行的预置字体栈供使用。



第 2 章 边框和图片

边框和图片是网页中不可或缺的元素，图片能够引起访客的兴趣，从而提高转化率。边框则可以使网页中各部分区域的范围明确，使网页的结构更清晰，同时很多输入框和搜索框的边框也是必不可少的。本章将介绍网页设计中常见的边框和图片的某些设计方案。

本章主要涉及的知识点如下：

- 边框新属性的基础与实例
- 搜索框、微博发布框、拍立得效果框等常用边框
- CSS 3 动画边框、边框移动特效
- Banner 图片的标签
- 黑白图片、图片水印
- 图片细节放大展示
- 瀑布流、图片墙、图片轮播图（焦点图）、幻灯片（带缩略图）
- 手风琴效果
- 纯 CSS 绘制图像
- 图片自适应、图片原地放大、图片翻转
- 图像地图

说明：考虑到动态变化效果，本章有不少实例使用了 JavaScript 代码来控制 CSS 样式，读者如果没有 JavaScript 基础也没有关系，实例代码会有详细的注释。

2.1 边框新属性的基础与实例

本节主要列出 CSS 3 增加的 border-color、border-image、border-radius 和 box-shadow 属性，CSS 3 中新增的边框属性使得 CSS 可以更强地控制边框样式。

2.1.1 border-color

使用 CSS3 的 border-radius 属性，如果边框的宽度是 X 像素，那在这个边框上可以使用 X 种颜色，每种颜色显示 1 像素的宽度。如果边框的宽度是 10 个像素，可以设置 10 种颜色，但是如果只声明了 5 或 6 种颜色，那么最后一个颜色将被添加到剩下的宽度。这一属性就如同渐变属性，在实际应用中使用较少，因为该属性仅在 Firefox 3.0+ 上得到支持。

2.1.2 border-image

border-image 是 CSS 中一个很强大的属性，也是一个让新手感到头痛的属性，因为它参数多而且复杂，而且在不同浏览器下得到的结果也不同。border-image 属性把边框的背景设置为图片，图 2.1 所示是该属性的一种应用，左边的部分为该属性所使用的原始图片，右边部分为浏览器使用原始图片处理之后的结果，看起来像是把原始图片拉长的效果。说到这里似乎有点名不符实，明明是给边框加图片，怎么看起来是背景图的效果，这个问题暂且交给读者思考，继续往下阅读，疑问自然会解开。



图 2.1 border-image 应用效果

在解释之前有必要对 border-image 属性的基础知识做一下认识。
在使用 border-image 前，必须要了解浏览器对它的支持，参见表 2.1。

表 2.1 浏览器对border-image属性的支持情况

浏览器及版本	是否支持及如何支持
Firefox 3.5- Firefox15（暂且称为老版本）	需加-moz-前缀
Firefox 15以上	同样支持-moz-前缀的CSS代码，但是存在bug，必须在CSS代码中加入border-style: solid;，否则不会看到效果
Chrome 1.1.X以上（老版本）	需要-webkit-前缀
Safari 3.1以上	需加-webkit-前缀
Opera浏览器（老版本）	需加-o-前缀
较新版本的Chrome、Firefox及webkit内核的opera	支持W3C标准，可以不加前缀
IE浏览器	支持效果不好，IE 11可以支持

border-image 的语法：
border-image:none | <image> [<number> | <percentage>]{1,4} [/ <border-width>{1,4}]? [stretch | repeat | round]{0,2}

单看语法便可知属性的复杂性。为了更清楚地讲述，下面使用更清晰的分类：

- 第 1 种写法 border-image:none;这种写法下边框不使用图片。
- 第 2 种写法的参数为表 2.2 中的 5 个参数，这 5 个参数可以只取参数值合起来写在 border-image 属性里，作为简写属性。也可以不使用 border-image 属性，直接使用下面的参数名：取值分开写 5 部分，通常使用简写属性。

表 2.2 更清晰分类下的 5 个参数

参数名称	参数描述（都无须单位px）	参数顺序
border-image-source	图片的路径	无
border-image-slice	图片的裁切方式，4个参数值可以是像素值，也可以为百分比，参数值可写一部分	裁切图片的上、右、下、左
border-image-width	边框的宽度，默认值存在边框宽度	上边框、右边框、下边框、左边框

续表

参数名称	参数描述（都无须单位px）	参数顺序
border-image-outset	边框偏移基准位置的像素值，默认值为0	上、右、下、左
border-image-repeat	使用裁切后图片填充的方式，可选值stretch repeat round，分别为拉伸、重复、平铺，默认值为stretch	上下边框、左右边框

列出分类的目的是为了更好地理解，而不是直接把参数名称写在 `border-image` 里。另外，中间带有参数顺序的 3 个参数之间需要使用斜杠（/）分开。例如下面的写法：

```
border-image: url(1.png) 30% 30% 30% 30% /5 5 5 5 stretch stretch ;
```

下面详细讲解 `border-image` 属性的使用。

（1）`border-image-slice`（图片裁切）详解

从现在开始，更换一幅经典的图片以便理解图片的裁切方式。图 2.2 左部分是原始图片，原始图片的大小为 78px×78px，右部分是以像素值为 26 像素的 4 个参数的切割示意图，这么切割之后，得到了一幅“九宫格”图，这幅九宫格图与边框的每个部分（图 2.3）又是吻合的。如果边框是有一定宽度的，那么四条边和与之包含的内容的组合也恰巧是一个“九宫格”，裁切完成的每部分小图片用来填充相应边框的部分。

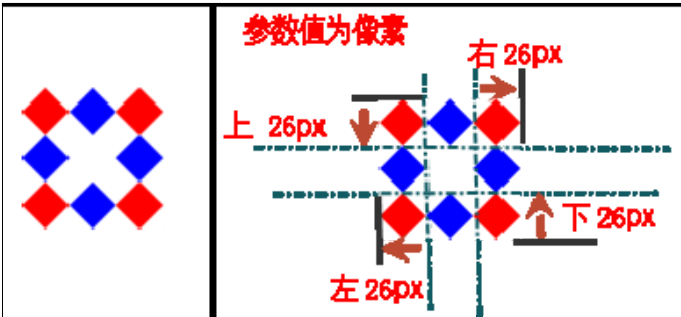


图 2.2 像素切割效果

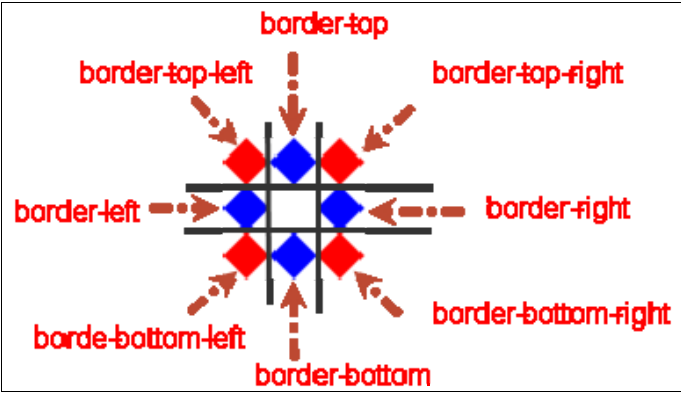


图 2.3 边框各部分示意图

除了使用像素，还可以使用百分比作为 4 个参数，使用百分比裁切的效果如图 2.4 所示，与使用像素值裁切是相似的。

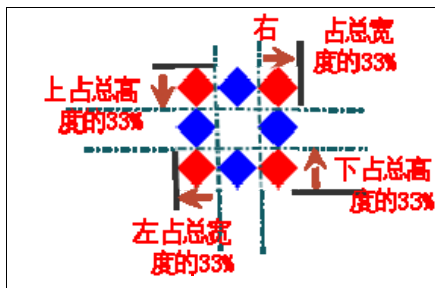


图 2.4 使用百分比裁切示意图

如果把属性分开写，上述的效果体现在代码上为：

```
border-image-slice: 26px 26px 26px 26px;或者 border-image-slice: 26px;
border-image-slice: 33% 33% 33% 33%;或者 border-image-slice: 33%;
```

(2) border-image-outset 详解

假设图 2.5 中的实线部分为边框应该在的基准位置，虚线部分为浏览器解释参数之后实际的边框开始位置，那么箭头所指的两线之间的距离即为 `border-image-outset` 的值，默认情况下值为 0，设置参数时按照上、右、下、左的顺序依次写出，并使用斜杠与上面的 4 个参数分开。

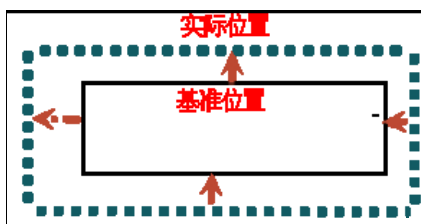


图 2.5 border-image-outset 详解图

默认情况下，浏览器内部的取值为：

```
border-image-outset: initial;等同于 border-image-outset: 0;
```

(3) border-image-repeat 详解

在上面提到了裁切之后每部分小图片对应相应的边框区域并且填充它们。在填充时，如图 2.6 所示，1、2、3、4 这几个部分的图片是不变化的，其余的部分是需要变化的。`border-image-repeat` 有两个参数，前一个参数影响的是 5、7 两部分对 top、bottom 边框的填充方式，应用的是水平填充规则；后一个参数这是影响 6、8 两部分对 right、left 边框的填充方式，应用的是竖直填充规则。第 9 小块则会在水平和竖直方向都应用填充规则。

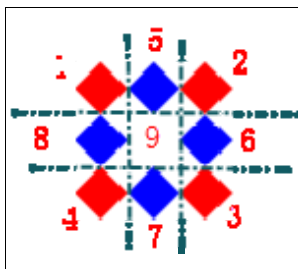


图 2.6 裁切得到的各部分

图 2.7 分别是拉伸、重复和平铺所对应的效果图，千言万语不及一张图，相信效果图能增强理解。

```
border-image: url(2.png) 26 26 26 26/10 stretch stretch ;
```



图 2.7 水平拉伸、竖直拉伸

```
border-image: url(2.png) 26 26 26 26/10 repeat stretch ;
```



图 2.8 水平重复、竖直拉伸

```
border-image: url(2.png) 26 26 26 26/10 repeat round ;
```

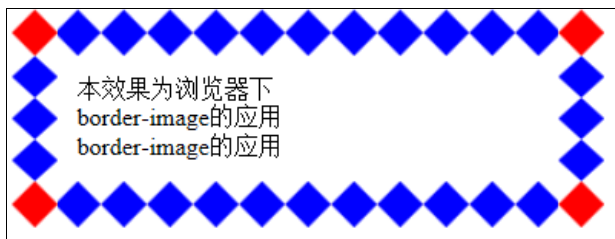


图 2.9 水平重复、竖直平铺

属性分开写的格式类似于：

```
border-image-repeat: stretch;
```

注意：重复是不改变裁切所得图片的大小，而平铺会自动改变图片的大小以达到合理的效果。

现在回到刚开始留下的问题，border-image 本意是为边框加图片，为什么变成了为元素加背景？原因在于，第一个案例中的图片裁切之后，9 处的位置并不是空白的，而是有一块蓝色的区域，该区域因同时应用水平填充和竖直填充，所以填充完成之后仍然为蓝色，

于是成为了背景。恰巧可以利用这一点制作可以伸缩的按钮图片，以解决不同大小的按钮需要准备不同大小的背景图片的问题。如果仍然想在中部留成空白，可以调整各个参数的值包括如何裁切、边框宽度、边框偏离基准位置的距离，以及元素的 padding 值来达到所需的效果。

注意：由于早期的带前缀的属性在各个浏览器上的支持并不完善，所以会出现相同的参数在使用前缀和不加前缀时得到的效果不同。这时建议把不加前缀的 CSS 代码放在后面，更标准的浏览器会使用后出现的值、忽略先出现的值。

在调试时，如果各部分参数把握不准，可以使用浏览器的控制台动态改变 CSS 代码，每次修改都会立即看到效果，从而更快地取得合理的参数值。

在 Firefox 下，如果看不到边框，不要忘记检查 border-style: solid;是否已经添加在 CSS 中。这是解决 Firefox 下 bug 的最快捷的办法。

2.1.3 border-radius

CSS 3 中新增了 border-radius 属性处理边框的圆角，在此之前，对边框圆角的处理大多是使用边框或者使用对多个小 div 的堆砌与定位来实现（该实例为本节源代码文件夹中的“CSS 2 圆角边框”，由于内容老旧，这里不做讲解）。而现在，除了使用 border-image 制作圆角边框，还可以使用 border-radius 属性轻松地设计出多种多样的圆角样式，需要的 CSS 代码仅仅为 1 行，还有很重要的一点：大多数浏览器对此属性的支持很好，表 2.3 为浏览器对该属性的支持情况。

表 2.3 浏览器对border-radius属性的支持情况

浏览器及版本	是否支持及如何支持
Firefox 3.0+（某些老版本）	需加-moz-前缀
Chrome 1.1.X+（老版本）	需要-webkit-前缀
Safari 3.1以上	需加-webkit-前缀
Opera浏览器（某些老版本）	需加-o-前缀
较新版本的Chrome、Firefox、Opera、Safari	可以不加前缀
IE 9+	不加前缀

该属性的语法为：

```
border-radius:none|<length>{1,4}[/<length>{1,4}]
```

在参数表中，length 可以选用 1~4 个像素值，[]中的内容可以省略，不省略时加 “/” 与前面的参数分开，实质上 4 个像素值参数最终转化为 border-top-left-radius、border-top-right-radius、border-bottom-right-radius 和 border-bottom-left-radius 的参数值，具体的对应情况见表 2.4 和表 2.5。

表 2.4 参数值转化情况表一

参数示例border-radius	5px	5px 6px	5px 6px 8px	5px 6px 8px 10px
border-top-left-radius	5px	5px	5px	5px
border-top-right-radius	5px	6px	6px	6px
border-bottom-right-radius	5px	5px	8px	8px
border-bottom-left-radius	5px	6px	6px	10px

表 2.5 参数值转化情况表二

参数示例border-radius	5px /5px	5px 6px /5px 6px	5px 6px 8px /5px 6px 8px	5px 6px 8px 10px /5px 6px 8px 10px
border-top-left-radius	5px 5px	5px 5px	5px 5px	5px 5px
border-top-right-radius	5px 5px	6px 6px	6px 6px	6px 6px
border-bottom-right-radius	5px 5px	5px 5px	8px 8px	8px 8px
border-bottom-left-radius	5px 5px	6px 6px	6px 6px	10px 10px

也就是说，当 CSS 代码为：

```
border-radius:5px 6px 8px 10px/5px 6px 8px 10px;
```

浏览器将按照：

```
border-top-left-radius:5px 5px;  
border-top-right-radius:6px 6px;  
border-bottom-right-radius:8px 8px;  
border-bottom-left-radius:10px 10px;
```

来解释。由表中的情况可知，当制作部分圆角效果时，参数将按照连续顺序写，如果只写 3 个参数，将对应 border-top-left-radius、border-top-right-radius 和 border-bottom-right-radius 解释，而不会是 border-top-left-radius、border-top-right-radius 和 border-bottom-left-radius，如果几部分的圆角是隔开的，那么可以分开写多条语句，或者在不需要圆角的地方使用参数 0，因为一旦参数中有 0 出现，就不再成为圆角而是直角。

图 2.10 以 border-radius:0 45px 0 0/0 60px 0 0 为实例，解释浏览器对圆角渲染的效果图。在表现效果上，这与 border-top-right-radius:45px 60px 是相同的。可以按照顶部边框上距离右上角 45 像素处的点与右边框上距离右上角 60 像素的点所画成的圆角来理解。

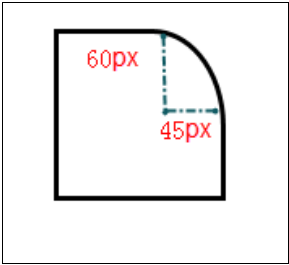


图 2.10 浏览器对圆角的渲染

2.1.4 box-shadow

在没接触 box-shadow 之前，可以使用 border-image 制作边框阴影的效果。box-shadow 的作用是向边框增加阴影效果，浏览器对其兼容性大致与 border-radius 相同，在某些需要体现层次感或是立体感的场景中常常会使用。其语法较为简单：

```
box-shadow: h-shadow v-shadow [blur spread color inset];
```

参数依次为水平方向偏移距离、垂直方向偏移距离、阴影模糊距离、阴影尺寸、阴影颜色、内阴影或是外阴影。其中前两个偏移距离参数必须设置，后面 4 个参数（中括号只是代表取值可选）可选。颜色默认值为黑色，inset 为内阴影、outset 为外阴影，默认情况

下是外阴影。这与文字与字体章节中文本阴影的参数相似。

`box-shadow:5px 8px 5px 3px red` 与 `box-shadow:5px 8px 5px 3px red inset` 的效果不同如图 2.11 所示，有关参数的说明也在其中。

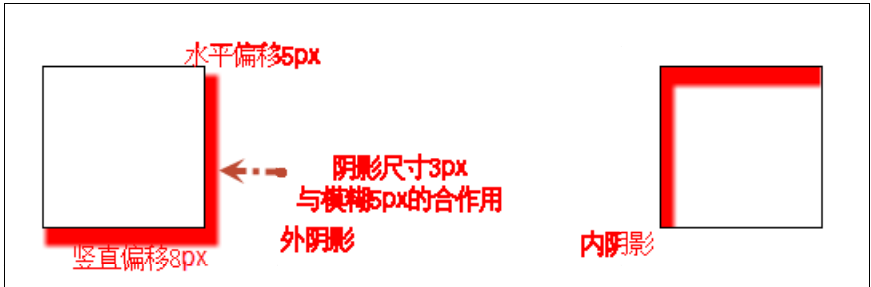


图 2.11 box-shadow 参数解释

注意：与 Photoshop 中阴影的设置方法一样，同一边框上可以使用多个投影，每两个投影之间以逗号隔开。

至此边框的某些新属性已经描述完毕，边框属性的应用会出现在不同的章节与实例中。

2.2 搜索框

本节旨在制作一个简洁却不失美感的搜索框。笔者使用两种思路对搜索框美化：一种是使用背景图片，另一种是只用 CSS。对比效果如图 2.12 所示。

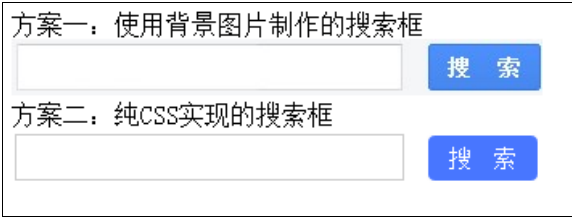


图 2.12 简洁的搜索框

2.2.1 使用背景图片的搜索框

这种方案为假美化，输入框与搜索按钮实质为背景图片中的影像。在设计时，使背景图片中的输入框和搜索按钮与实际输入框和搜索按钮的位置重合，在完成输入框和搜索按钮的尺寸、文本的调整之后，使得一切边框皆为 0、一切挡住背景图片的元素均透明即可。使用这种方案可以通过更换背景图片方便快捷地更换搜索框的样式，也可以实现普通 CSS 难以实现的复杂搜索框效果（例如搜索框的周围布置上花藤）。

```
//HTML 部分
01 <div id="search">
02 <input type="text" id="input" />
```

```

03 <input type="button" id="go" />
04 </div>

//CSS 部分
01 #search
02 {
03     position:relative;
04     padding-top:3px;           //输入框向下偏移到合适高度
05     height:35px;             //长和宽最好与背景图一致
06     width:330px;
07     background:url(search.jpg) no-repeat; //背景图
08 }
09 #input
10 {
11     left:2px;                 //向右偏移到合适的位置
12     position:absolute;
13     font-size:20px;
14     width:240px;              //长和宽与背景图有关
15     height:25px;
16     border:0;                 //隐藏默认属性
17     background:none;
18 }
19 #go
20 {
21     position:absolute;
22     left:258px;
23     width:70px;               //长和宽与背景图有关
24     height:30px;
25     cursor:pointer;
26     border:0;                 //隐藏默认属性
27     background:none;         //隐藏默认属性
28 }

```

以上代码为实例，最主要的工作是定位方面，如代码中第 4 行、第 7 行、第 22 行调整元素与背景图片重合。第 16 行、第 17 行、第 26 行、第 27 行则是把遮盖背景图的边框、输入框背景等元素变为透明元素。

2.2.2 只使用 CSS 的搜索框

对于本示例中的效果来说，完全可简单地用颜色值、圆角、大小、定位来实现与方案一相同的效果（图 2.12 下部分）。纯 CSS 代码不受图片的限制，响应快、效果好。只是不好实现复杂样式的搜索框。

```

//CSS 部分
01 #search2

```

```

02 {
03     position:relative;
04     padding-top:3px;
05     height:35px;
06     width:330px;
07 }
08 #input2
09 {
10     left:2px;
11     position:absolute;
12     font-size:20px;
13     width:240px;
14     height:25px;
15     border:0;
16     background:#FFFFFF;           //颜色
17     border:1px solid #CCCCCC;     //颜色
18 }
19 #go2
20 {
21     font-size:14px;
22     color:#FFFFFF;
23     position:absolute;
24     left:258px;
25     width:70px;
26     height:30px;
27     cursor:pointer;
28     border:0;
29     background:#4876FF;           //颜色
30     border:1px solid;
31     border-radius:5px;            //圆角
32 }

```

代码较为简单，不再赘述。

注意：当更换背景图片、添加输入框发光效果、输入框背景色渐变、添加 JavaScript 代码时，可带来样式改变、立体效果增强和输入框动画等效果，所以搜索框样式可多种多样，本书只列出简单搜索框美化，这更加适应扁平化设计的风格。

2.3 微博发布框

在微博流行的年代，微博发布框人们都不陌生，它不仅仅是一个输入框，还包括 140 字判断、提交判断等功能。新浪微博、腾讯微博的功能几乎一致，其他类似于 QQ 空间说说发布框的模块也可以借鉴。本节的实例主要实现仿腾讯微博发布框，主要内容为 CSS 定

位布局、JavaScript 完善功能。图 2.13 为微博发布框的效果图。



图 2.13 微博发布框

//HTML 代码

```
01 <div class="weibodiv">
02 <a href="#" class="ad"></a>
03 <a href="#" class="adtext">点击牛运按钮，赢取码上好礼</a>
04 <div class="weibotext">
05 <textarea></textarea>
06 <p id="weibotextnum">还能输入<span id="weibotextnumber">140</span>字</p>
07 </div>
08 <span id="weibobottomlinks"></span>
09 <a class="post">广播</a>
10 </div>
```

在本实例的 HTML 中，发布框下方的链接并没有做，而是用 img 代替。在 CSS 部分，主要分为 3 块：上部、中部和底部。上部大多放置广告图和广告链接或是活动链接。左边元素使用左浮动、右边元素使用右浮动即可。在输入框上，应用输入框发光效果。字数提示默认不可见，只有在输入文字时可见。CSS 部分的代码如下：

```
01 *{margin:0;padding:0;font-size:12px;}
02 .weibodiv{
03     width:600px;
04     margin:20px auto;           /*整体水平居中*/
05 }
06 .ad{                             /*上方广告模块*/
07     display:block;
08     width:291px;
09     height:30px;
10     background:url(img/1.png);
11     float:left;                 /*浮动在左边*/
12 }
13 .adtext{                         /*广告文本*/
14     position:relative;
15     float:right;                /*浮动在右边*/
16     line-height:30px;
```



```

17     top:10px;
18 }
19 p{
20     float:right;
21 }
22 .weibotext{
23     margin:5px auto;
24     float:left;
25     top:30px;
26 }
27 textarea{                                /*文本输入区*/
28     width:590px;
29     height:150px;
30     font-size:16px;
31     overflow:auto;
32 }
33 .post{                                    /*发布按钮*/
34     display:block;                        /*此区域的设置同第3章中按钮的设置方法*/
35     color:#FFFFFF;
36     float:right;
37     border:1px solid;
38     width:80px;
39     height:30px;
40     text-align:center;
41     text-decoration:none;
42     line-height:30px;
43     margin:3px;
44     border-radius:5px;
45     font-size:16px;
46     font-weight:bold;
47     letter-spacing:5px;
48     background:#8BC528;
49     cursor:pointer;
50 }
51 #weibotextnumber{                        /*字数统计区域*/
52     font-family:Bell MT;                /*效果较好的字体*/
53     font-size:20px;
54 }
55 #weibotextnum{                            /*字的数量*/
56     opacity:0;                          /*没有输入则隐藏*/
57 }

```

第4行使整个发布框居中，第14行和第17行调整右上角的文字高度，解决右浮动后文字处于右上角的问题。提交按钮的CSS样式较多，包括设置圆角、颜色、文字几何居中、文字间隔等。CSS代码较为简单，主要是基本样式和定位即可。

在 JavaScript 部分需要做的工作有：

- 输入框获得焦点时文本框发光——使用 onfocus 事件改变 CSS 样式。
- 字数提示显示——使用 onpropertychange 和 oninput 事件，在输入框字数发生改变时使用正则表达式统计出当前已经输入的字数并做判断，同时使用 JavaScript 写入到 HTML 中给用户当前字数的提示。
- 输入框失去焦点时输入框回复初始状态——使用 onblur 事件改变 CSS 样式。
- 输入字数时判断字数改变提示文字的具体实现。
- 提交时判断字数是否符合要求。

以下的 JavaScript 代码为一种可行的方案：

```

01 <script type="text/javascript">
02 window.onload =function (){
03   var oT = document.getElementsByTagName('textarea')[0];           //获取到输入区
04   var weibotext=document.getElementsByClassName("weibotext")[0]; //获取输入区外容器
05   var weibotextnum=document.getElementById("weibotextnum");       //获取数字提示语句
06   var weibotextnumber = document .getElementById('weibotextnumber'); //获取数字
07   var oA=document.getElementsByClassName('post')[0];             //获取提交按钮
08   var ie = !-[1,];                                                //判断是否为 IE
09   oT.onfocus=function (){                                         //获得焦点函数
10     oT.style.border="1px #40E0D0 solid";
11     oT.style.boxShadow="0 0 10px #5CACEE";
12     weibotextnum.style.opacity="1";
13     var num = Math.ceil(getLength(oT.value)/2);
14     if(num=="")
15     {
16       oA.style.background="#7F7F7F";                             //第一次获得焦点提交按钮为灰色
17     }
18   };
19   oT.onblur = function(){                                          //失去焦点函数
20     oT.style.boxShadow="";
21     weibotextnum.style.opacity ="0";
22     oA.style.background="#8BC528";
23   };
24   if(ie){                                                          //字数变化事件的兼容,主要针对 IE
25     oT.onpropertychange = toChange;
26   }
27   else{
28     oT.oninput = toChange;
29   }
30   oA.onmouseover=function(){
31     oT.blur();
32     oA.style.background="#7CCD7C";
33   }

```

```

34     oA.onmouseout=function(){
35     oA.style.background="#8BC528";
36     }
37     oA.onclick=function (){           //提交函数
38     var num = Math.ceil(getLength(oT.value)/2);       //获取内容长度
39     if(num==0|| num>140){
40         alert("不符合发布要求，请检查");
41     }
42     else{
43         alert("发布成功");           //实际中动态加载到页面上
44         oT.value="";
45         weibotextnumber.innerHTML = "140";
46     }
47     }
48     function toChange(){               //字数变化时执行的函数
49         var num = Math.ceil(getLength(oT.value)/2);
50         if(num<=140){
51             weibotextnum.innerHTML =
52             "还能输入<span id='weibotextnumber'></span>52  字";
53             weibotextnumber = document .getElementById('weibotextnumber');
54             //重新获取数字
55             weibotextnumber.innerHTML = 140 - num;
56             weibotextnumber.style.color = "";
57         }
58         else{
59             weibotextnum.innerHTML = "超出<span id='weibotextnumber'></span>字，您
60             可以转为<a href='#'>长微博</a>发送";
61             weibotextnumber = document .getElementById('weibotextnumber');
62             weibotextnumber.innerHTML=num-140;
63             weibotextnumber.style.color = 'red';
64         }
65     }
66     if(oT.value==" " || num>140){
67         oA.style.background="#7F7F7F";    //提交框为灰色
68     }
69     else{
70         oA.style.background="#8BC528";    //提交框为绿色
71     }
72     }
73     function getLength(str){             //计算输入内容长度的函数
74     return String(str).replace(/[\^\x00-\xff]/g,'aa').length;//正则表达式，当汉字时以 aa 算长度
75     }
76 }
77 </script>

```

2.4 拍立得效果框

拍立得效果框多用来装饰照片，使用 border 或元素边距、盒子阴影即可创建拍立得效果，图 2.14 为拍立得效果图。使用 padding 制作最简单快捷，如果不需要在图片上加文字说明，仅用 border 也可实现。

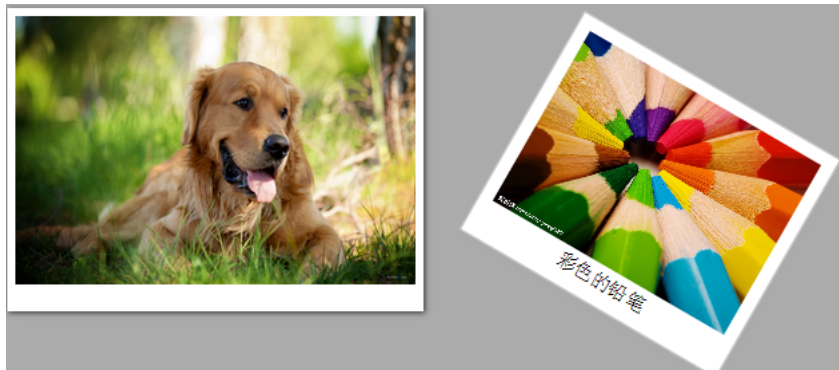


图 2.14 拍立得效果

CSS 代码如下：

```
01 body
02 {
03     background:#ABABAB;
04 }
05 img.palaroid{
06     background:#000;
07     height:200px;
08     width:300px;
09     border:solid #fff;
10     box-shadow:1px 1px 5px #333;          /*边框阴影*/
11     border-width:6px 6px 20px 6px;        /*各部分边框有不同的宽度*/
12 }
```

第 11 行使各部分的边框宽度不同，第 10 行为盒子加阴影。如果需要为图片添加说明文字，那么用 div 是最简单的：

```
01 div.palaroid
02 {
03     position:absolute;
04     left:50px;
05     top:300px;
06     text-align:center;
07     font-size:15px;
08     width:294px;
09     padding:10px 10px 10px 10px;          /*使用 padding*/
10     border:1px solid #BFBFBF;
```

```

11     background-color:white;
12     box-shadow:2px 2px 3px #aaaaaa;
13     transform:rotate(30deg);                /* 旋转*/
14     -ms-transform:rotate(30deg);            /* IE 9 */
15     -moz-transform:rotate(30deg);          /* Firefox */
16     -webkit-transform:rotate(30deg);       /* Safari and Chrome */
17     -o-transform:rotate(30deg);           /* Opera */
18 }

```

第13~17行向div应用了CSS 3中的2D转换,使div绕其几何中心顺时针旋转30度。有关2D转换的内容,可以参考动画基础的章节。这种应用在图片墙中也是必须设置的效果,在此基础上实现的图片墙可参照后面的章节。

2.5 CSS 3 动画边框

在网页设计中经常使用边框使得区块分明,不管边框是什么形状,也不管边框有多美观,边框大多是静态的。在CSS 3时代,完全可以不借助JavaScript实现动画的效果,因为CSS 3动画即可胜任。本节将用CSS 3动画和背景巧妙实现图片的动画边框,效果如图2.15所示,图片是静态的,实际上边框是不断变化的。它的本质并不是边框而是背景,所以它是“伪边框”,从视觉上看却根本察觉不出。这也是本实例的巧妙之处。



图 2.15 动画边框

因为是“伪边框”,所以图片下面是有斜纹的,只是被图片挡住了,HTML部分是:

```

01 <div id="demo">
02     <div class="animation_border"></div>
03     
04 </div>

```

外层的div作为容器,内层的div作为背景容器,这里选取两个div的原因是后面动画中将使用透明度设置,默认边框是显示不出来的,只有在鼠标指向图片时边框才会出现,如果仅使用1个div,透明度为0时会造成图片不可见。这里不需要对只有HTML代码的网页做演示,直接开始应用CSS:

```

01  *{
02      margin:0;           /*简单初始化*/
03      padding:0;
04  }
05  #demo {                 /*图片下方边框预留区域定位*/
06      width: 300px;
07      padding: 10px;
08      position: absolute;
09      left:200px;
10      border:1px solid;
11  }
12  #demo > img {           /*图片设置*/
13      display: block;
14      position: relative;
15      cursor:pointer;
16  }

```

第 1~4 行把外边距和内边距重置为 0，第 5~11 行对容器大小和位置做设置。第 7 行把容器的内边距设置为 10 像素，这里是边框的存在区。第 12~16 行使图片块状显示，指向图片的鼠标样式为手形。执行以上的 CSS 代码，会得到图 2.16 所示的效果。



图 2.16 初步设置

下面不会再对图片做出改变，暂且把图片放在一边。开始“伪边框”——背景的制作。首先应该让内层 div 充满外层 div。

```

01  .animation_border {
02      position: absolute;
03      top: 0;
04      left: 0;
05      right: 0;
06      bottom: 0;
07  }

```

为了讲述方便，笔者首先把背景设置为正方形，然后利用 CSS 渐变做一个斜纹的背景，以下代码执行之后的效果如图 2.17 所示。

```

01  .animation_border {
02      background-image: -webkit-linear-gradient(45deg, red 25%, transparent 25%,
03          transparent 50%, red 50%, red 75%, transparent 75%, transparent);
04      background-image: -moz-linear-gradient(45deg, red 25%, transparent 25%, transparent

```

```

05      50%, red 50%, red 75%, transparent 75%, transparent);
06      background-image: linear-gradient(45deg, red 25%, transparent 25%, transparent 50%,
07      red 50%, red 75%, transparent 75%, transparent);
08  }

```

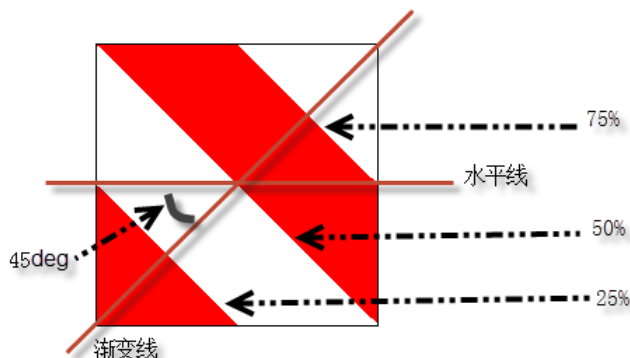


图 2.17 渐变斜纹图

CSS 渐变的第一个参数选用了度数，在`-webkit-`前缀、`-moz-`前缀和`-o-`前缀下，这一度数是由水平线沿逆时针方向与渐变线的角度，如图 2.17 所示。在没有前缀时，这一度数是由竖直线沿逆时针方向与渐变线的角度。其他的颜色暂停点都已经在图中标注出来，从 0%~25%和 50%~75%是红色，25%~50 和 75%~100%是透明的，也就是背景白色。

注意：谷歌浏览器、IE 浏览器、Opera 浏览器最新版本都支持不加前缀，书写前缀是为了兼容火狐浏览器、Safari 浏览器和许多低版本浏览器。

这样看来，虽然现在已经借助 CSS 渐变实现了斜纹，但是这一大图做背景显然是不合适的，并且达不到很多条纹的效果。如果细分更多的百分比制作更多条纹，显然也是不合适的。这时候背景图片的大小就很有用了，再添加一行代码。执行后的结果如图 2.18 所示。

```

01 .animation_border {
02     background-size: 30px 30px;
03 }

```

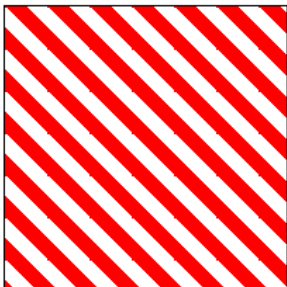


图 2.18 背景图片大小改变之后

下一步便是如何让条纹动起来，需要使用 `background-position` 属性，图 2.19 所示的条纹运动原理图中，当背景图由位置 1 缓慢地变化到位置 2，即 `background-position` 的值由 0, 0 变化到 60, 30 时，从视觉上看，条纹就会动起来。实现运动的数值不是固定的，60, 30

仅为本例中一种可行的方案，其他合理的数值也是可以的。为了加强效果原理的理解，建议读者截取两幅同样大小的条纹图，做拖动条纹图从一个位置到另一个位置的实验，理解就不难了。

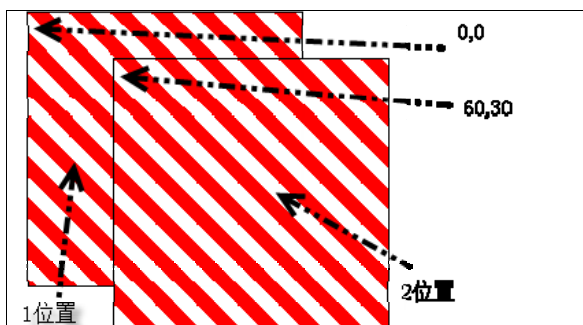


图 2.19 条纹运动原理图

得到合适的值之后，下一步与 CSS 3 动画结合起来，首先使用@keyframes 规则定义适应各种浏览器的动画。然后让斜条纹执行这些动画。下面是这部分的 CSS 代码，执行这些代码之后，斜条纹就可以运动了，这时把图片显示出来，把条纹的大部分遮盖起来，只剩外层容器的 padding 部分是图片遮盖不住的内容，这样看来，斜纹背景便成为图片的运动边框了。

```

01  @-webkit-keyframes animate {                /*定义动画 webkit*/
02  from {
03      background-position: 0 0;
04  }
05  to {
06      background-position: 60px 30px;
07  }
08  }
09  @-moz-keyframes animate {                   /*定义动画 opera*/
10  from {
11      background-position: 0 0;
12  }
13  to {
14      background-position: 60px 30px;
15  }
16  }
17  @keyframes animate {                       /*定义动画 w3c*/
18  from {
19      background-position: 0 0;
20  }
21  to {
22      background-position: 60px 30px;
23  }
24  }

```



```

25 .animation_border {                               /*应用动画*/
26     -webkit-animation: animate 0.5s linear infinite;
27     -moz-animation: animate 0.5s linear infinite;
28     animation: animate 0.5s linear infinite;
29 }

```

最后，使运动的斜纹背景设置为透明，当鼠标指向图片时，通过：`hover` 伪类将其透明度改变使得边框展现。为了加强视觉体验，再使用 CSS 3 中的 `transition` 属性为动画做缓冲的效果，鼠标指向图片时在 0.3s 内将边框缓慢展示，鼠标移开时淡出，代码如下。

```

01 .animation_border {
02     opacity: 0;
03     -webkit-transition: opacity 0.3s ease;
04     -moz-transition: opacity 0.3s ease;
05     transition: opacity 0.3s ease;
06 }
07 #demo:hover .animation_border{
08     opacity: 1;
09 }

```

至此本实例的所有代码均已完成，完整的代码请参考实例。

注意：本实例运用的是转换的思想，即通过设置背景转换为边框，用法巧妙并且可以扩展到其他方面，如会动的进度条。

在 CSS 的渐变部分，可以使用重复渐变而不必使用 `background-size` 实现，具体的实现可以参照渐变（CSS 背景和颜色基础）的相关章节。

本节 CSS 部分同时涉及了第 5 章的内容，如有疑问可详细查看第 5 章基础知识一节。

2.6 边框移动特效

在迅雷的登录界面上，在用户名和密码输入框上切换时有一款很漂亮的边框移动特效，本节的实例就是把迅雷登录界面的这一特效扩展到网页中，相信这一特效能够带来良好的用户体验。效果图如图 2.20 所示，当点击某个输入框或者按 `Tab` 键使焦点框获得焦点时，边框就会移动到该输入框上，同时尺寸也会变化到该输入框的大小。

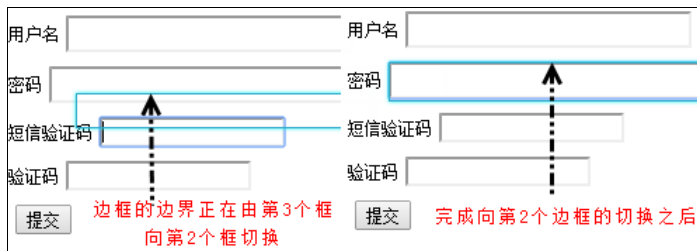


图 2.20 边框移动特效

实现的原理非常简单，在网页中有许多 input 框和一个 div，这个 div 为绝对定位。看到的边框移动实际上是 div 大小和位置改变的结果，并不是输入框的边框到处飞。所以在 CSS 部分只需要对此 div 进行设置。

```
01 #animate_border{
02     position:absolute;left:0;top:0;
03     width:0;height:0;border:1px solid #2CAAC9;box-shadow:0 0 3px #0AF1F5;
04 }
```

在 JavaScript 部分，为每个输入框添加 onfocus 事件，当输入框获得焦点时，获取到该输入框的长度、宽度、左偏移值和上偏移值，并将其应用在 div 的 length、width、left 和 top 属性上，当然要使 div 均匀变化到目标值（也就是运动）。

一种 JavaScript 解决方案如下，运动部分同样使用本书经常用到的运动框架（框架函数见源代码）：

```
01 <script type="text/javascript" src="js/zQuery.js"></script>
02 <script type="text/javascript">
03 window.onload=function(){
04     var inputs=document.getElementsByTagName('input');           //获取到输入框
05     var animate_border=document.getElementById('animate_border'); //获取到 div
06     for (var i=0;i<inputs.length;i++)
07     {
08         //为输入框添加 onfocus 事件并传入变化函数
09         inputs[i].onfocus=function(){change_border_to(this);}
10     }
11     function change_border_to(obj){                               //出入某个输入框
12         var this_width=obj.offsetWidth;                           //获得该输入框的宽度
13         var this_height=obj.offsetHeight;                           //获得该输入框的高度
14         var this_left=obj.offsetLeft;                               //获得该输入框的左偏移值
15         var this_top=obj.offsetTop;                                 //获得该输入框的上偏移值
16         move(animate_border,{width:[this_width],height:[this_height],left:[this_left],top:[this_top]});
17         //使用运动框架动态改变 div 样式
18     }
19 }
20 </script>
```

注意：运动框架的功能与 jQuery 中的 animate 效果相似，可以轻易地根据本实例中的注释扩展同样的效果到 jQuery 上。

2.7 Banner 图片的标签

使用 CSS 代码可以在 Banner 图片上添加一个标签效果，看起来如图 2.21 所示。

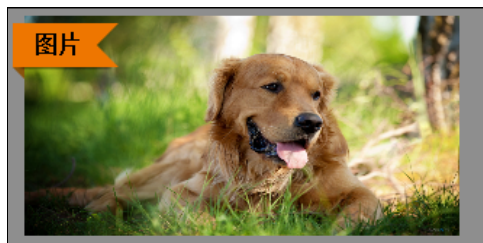


图 2.21 标签效果图

在开始之前，首先说明一下如何用 `border` 制作各式各样的形状，在许多导航链接中使用一个小三角形指明这是一个下拉菜单，如果该处也要使用一张图片，必然麻烦了点。使用 `border` 即可轻松创建，一些图形如图 2.22 所示，当然也可以制作梯形。

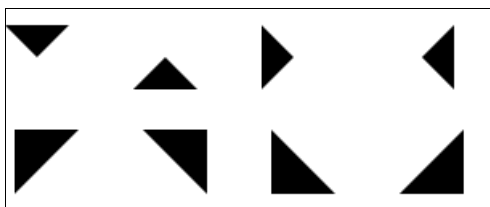


图 2.22 使用 border 创建图形

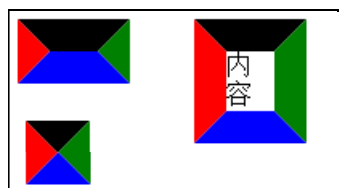


图 2.23 边框

在网页中，边框并不是四条线组合在一起的，当边框足够宽时，就能看到边框的真正面目，如图 2.23 所示。当使某些边框都变成透明时，各种形状组合就出现了。如：

```
01 border-top:20px solid;
02 border-left:20px solid transparent;
03 border-right:20px solid transparent;
04 border-bottom:20px solid transparent;
```

几张图对比就可以轻易发现实例的原理，使用 `before` 伪类在图片上方添加一个去掉右边框的元素，使用 `after` 伪类在图片下方添加一个三角形，并使用 `content` 添加文字。代码如下：

```
01 .featureBanner:before {
02     position:absolute;top:5px;left:-8px;z-index:1;           /*定位相关*/
03     padding-right:10px;font-weight:bold;line-height:0px;
04     color:#000;height:0px;
05     border: 15px solid #EE7600;                               /*利用的是 border 属性*/
06     border-right-color: transparent;                          /*去掉右边的边框*/
07     content: "图片";                                         /*这里定义标签上的文字*/
08     box-shadow:0px 5px 5px -5px #000;
09 }
10 .featureBanner:after {                                       /*第二部分的效果*/
11     content: "";
12     position:absolute;top:35px;left:-8px;
13     border: 4px solid #89540c;
14     border-left-color: transparent;
15     border-bottom-color: transparent;
```

```
16 }
```

在 CSS 3 的时代，网页设计中可以使用更多新属性将 div 打造成各式各样的图标，One div 就是一个专门使用纯 CSS 3 代码制作图标的一员，虽然这些图标存在低版本浏览器的兼容问题，但这并无大碍。

注意：可以使用 CSS 3 圆角、CSS 3 渐变、before 和 after 等属性创建回形针等其他效果，然后将回形针与拍立得效果框结合起来会有很好的效果。

2.8 黑白图片

有时在网页中需要使用黑白照片，比如在汶川地震时，各大网站都把自己的 logo 变为黑白图。早在 IE6~IE9 时代，黑白图片在 IE 下可以使用滤镜（filter: gray）轻松实现，但滤镜只是 IE 的“专利”，其他浏览器无法识别并且从 IE10 开始 IE 浏览器也抛弃了滤镜。在 CSS 3 中，理论上可以使用 filter: grayscale(100%)来实现，然而这一属性只能在 Chrome 浏览器和使用 Webkit 内核的 Opera 浏览器下通过设置前缀-webkit-来实现。

为了使效果兼容更多的浏览器，笔者推荐使用一个名为 grayscale.js 的 JavaScript 来实现，经测试在所有的浏览器下均正常工作。它的简洁的使用方法为：

```
01 var oimg=document.getElementsByTagName('img')[1];
02 grayscale(oimg);
```

只需把对象作为参数传入到 grayscale 方法中即可。

注意：实现黑白照片的途径并非只有一条。使用 JavaScript 是最简洁方便、兼容性好的方案。正如牛顿所说“站在巨人的肩膀上”，使用前人的经验往往使得解决问题变得简单，带来工作效果的提高。有的时候，需要从多种角度（资源、代码开销、复杂度等）综合考虑整体的代价然后制定解决方案。

2.9 图片水印

网站上使用图片水印一般是在图片上加上自己网站的网址、图标或者某些文字。此类应用多为防盗图而设置。如果图片加水印目的在于图片防盗，那么最根本的方式必然是直接用图片工具（如 Photoshop）处理了。但如果仅仅为了得到图片水印的效果，那么希望本节的实例能有用，实例中使用的是定位法将图片说明附到图片的某一位置，效果如图 2.24 所示。

使用纯 CSS 布局实现，代码很简洁，思路很简单。代码如下：

```
01 div.shui{
02     border:1px solid;opacity:0.5;
03     position:absolute;bottom:0;right:0;
04 }
05 div.img{border:1px solid ;position:relative;display:inline-block;}
06 img.big{width:250px;height:150px;}
```

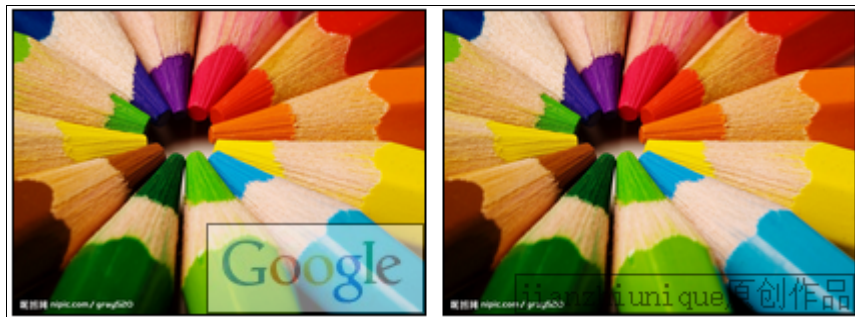



图 2.24 使用图片作为水印和使用文字作为水印的效果展示

注意：如果要尽可能地实现图片防盗，可以把标签用background替代。也可以通过在最上层加一个遮罩层来阻止普通用户下载图片（依然可以通过浏览器控制台获取到图片资源）。

2.10 图片细节放大展示

几乎在所有的电子商务网站（如当当、淘宝、京东商城）里，都有图片放大效果（也被称为放大镜）的应用。图片放大展示效果的主要目的是让顾客看到更清晰、更接近实物的大图，为购买提供便利。图 2.25 是本节实例的效果图，虽然各种网站上的图片展示效果代码不尽相同，但原理基本是相同的。



图 2.25 图片放大效果

在 HTML 中，对最外层

div

命名，便于在多处应用时取得整体元素。在此

div

中，包含两个小

div

，一个是左边小图所在的

div

，另一个是右边大图所在的

div

，赋给他们 CSS 类名以便重复利用。在左

div

中，还需要有一个与

div

大小一致的cover遮罩和一个半透明的小浮动层。综合起来的 HTML 代码如下：

```
01 <div id="show_bigger_pic">
02 <span class="cover"></span>
```

```

03 <span class="float_span"></span>
04 <div class="small_pic_div">
05 
06 </div>
07 <div class="big_pic_div">
08 
09 </div>
10 </div>

```

在布局 CSS 之前，首先应考虑整体。为了增强实用性以及减少出错的可能性，应该把最外层的定位设置为绝对定位，然后再做处理。如果绝对定位没有问题，其他的定位方式一般也能通过。大小图 div 应该为左浮动，小图 div 应为相对定位以便子元素进行绝对定位，小图 div 的子 span 元素都应为绝对定位脱离文档流。在初步布局做好之后，把大图 div 和浮动层都隐藏起来，考虑完成后再开始写 CSS 代码。

```

01 *{margin:0;padding:0;}
02 #show_bigger_pic{
03     position:absolute;
04     width:500px;
05     height:400px;
06     top:200px;
07     left:200px;
08 }
09 .small_pic_div{           /*小图区*/
10     width:240px;
11     height:240px;
12     border:1px solid;
13     float:left;
14     position:relative;    /*便于覆盖层的绝对定位*/
15 }
16 .big_pic_div{             /*大图区*/
17     width:240px;
18     height:240px;
19     border:1px solid;
20     float:left;
21     margin-left:10px;
22     display:none;         /*大图默认不显示*/
23     overflow:hidden;
24 }
25 .big_pic_div>img{
26     position:relative;
27 }
28 .cover{
29     width:240px;
30     height:240px;
31     position:absolute;    /*覆盖层绝对定位*/

```

```

32     border:1px solid;
33     z-index:2;
34     left:0;           /*绝对定位时不指定 left 和 top 在火狐下会产生错误*/
35     top:0;
36 }
37 .float_span{
38     width:165px;
39     height:165px;
40     position:absolute;
41     left:0;           /*绝对定位时不指定 left 和 top 在火狐下会产生错误*/
42     top:0;
43     z-index:1;
44     background:#B2DFEE;
45     opacity:0.5;
46     display:none;     /*当前放大区域指示器默认不显示，只在鼠标指向图片时显示*/
47     border:1px solid;
48 }

```

为了减少错误几率，第 3~7 行对整体 div 设置了随意的值。由于本实例中元素大小的值可由 JavaScript 部分动态决定，所以第 10~11 行、第 17~18 行、第 29~30 行和第 36~37 行对大小的设置可随意设置，但 cover 层和小图 div 的大小是一致的。cover 层的作用是供 onmousemove 事件使用，所以需要把整个小图都覆盖起来，并且它在重叠的多个元素的最上面（第 33 行）。第 21 行将左右的两部分设置间距，第 23 行将大图 div 之外的图片区域隐藏起来。

注意：理论上讲，使用图片作为背景图片，结合 background-size 和 background-position 属性也可实现，但在浏览器兼容方面不如使用 img 定位实现好，所以本例中使用的是 img 定位的方法。

在 JavaScript 方面，主要做的工作有以下几个方面，这些也是基本通用的流程。

- （1）在页面加载完成之后取得所有将要用到的 HTML 元素。
- （2）鼠标移动到小图上时，鼠标样式改变，右边大图出现，小图上的浮动层也出现，并在鼠标的位置。同时记录小图、大图、小 div 的大小参数供后面使用。
- （3）鼠标移开小图时，第 2 步的样式逆向改变。
- （4）鼠标在小图上移动时，获取到鼠标的位置，并转换为浮动层相对于小图 div 的定位 left 值和 top 值。转换方法为：left 值为鼠标的横坐标位置-最外层容器距离左边缘的位置-浮动层宽度的一半；top 值为鼠标的纵坐标位置-最外层容器距离上边缘的位置-浮动层高度的一半。并需要对浮动元素的可移动区做限制，当 left、top 值小于 0 或大于一定数值之后，将数值强制变为 0 或最大值从而使浮动层固定。
- （5）右面的大图的 left 值与 top 值应伴随浮动层的改变而改变。改变方法因为：left 值=比例*大图的宽；top 值=比例*大图高度。其中的比例分别为 percentX=left/小图宽度、percentY=top/小图高度。
- （6）为了使右图的展示区域符合左图中浮动层的覆盖区域，需要对浮动层大小进行计

算，计算使用第 2 步中的值，具体参考代码。

(7) 写 JavaScript，修复错误，增强代码重用性。

以下的 JavaScript 代码是一种可以实现的解决方案，对应上述的 JavaScript 说明并带有注释帮助理解。

```

01 <script type="text/javascript">
02 function gbc(tparent,tclass){           //获取指定父元素的指定类的子元素的函数
03     var allclass=tparent.getElementsByTagName("*");
04     var result=[];
05     for (var i=0;i<allclass.length;i++)
06     {
07         if(allclass[i].className==tclass)
08             result.push(allclass[i]);
09     }
10     return result;                       //返回的是数组
11 }
12 window.onload =function (){
13     var sbp=document.getElementById('show_bigger_pic'); //获取最外层 div
14     var c=gbc(sbp,'cover')[0];                          //获取 cover 层
15     var fs=gbc(sbp,'float_span')[0];                    //获取浮动层
16     var spd=gbc(sbp,'small_pic_div')[0];                //获取小图 div
17     var sp=spd.getElementsByTagName('img')[0];          //获取小图
18     var bpd=gbc(sbp,'big_pic_div')[0];                   //获取大图 div
19     var bp=bpd.getElementsByTagName('img')[0];          //获取大图
20     var spw;                                              //比例计算暂存参数变量
21     var sph;
22     var bpw;
23     var bph;
24     var btn=true;                                        //开关，因参数只需获取一次
25     c.onmouseover =function(){                          //鼠标移入小图
26         fs.style.display="block";
27         bpd.style.display="block";
28         c.style.cursor="pointer";
29         if(btn){                                         //获取大小图片的参数以便浮动层大小的计算，仅获取一次即可
30             spw=sp.offsetWidth;
31             sph=sp.offsetHeight;
32             bpw=bp.offsetWidth;
33             bph=bp.offsetHeight;
34             spd=sp.offsetWidth;
35             spdh=sp.offsetHeight;
36             var fsw=Math.ceil(spw/bpw*spd);              //比例计算
37             var fsh=Math.ceil(sph/bph*spdh);
38             fs.style.width=fsw+'px';                     //浮动层大小设置
39             fs.style.height=fsh+'px';
40             btn=false;                                   //获取完关闭开关

```



```

41     }
42 };
43 c.onmouseout =function(){                                //鼠标移出
44     fs.style.display="none";
45     bpd.style.display="none";
46 };
47 c.onmousemove =function (ev){                            //鼠标移动
48     var pos=ev||event;
49     var left=pos.clientX-sbp.offsetLeft-fs.offsetWidth/2;    //计算 left
50     var top=pos.clientY-sbp.offsetTop-fs.offsetHeight/2;    //计算 top
51     if(left<0){
52         left=0;                                            //当小于 0 强制固定
53     }
54     else if(left>c.offsetWidth-fs.offsetWidth){          //防浮动层移出图片区
55         left=c.offsetWidth-fs.offsetWidth;
56     }
57     if(top<0){
58         top=0;
59     }
60     else if(top>c.offsetHeight-fs.offsetHeight){
61         top=c.offsetHeight-fs.offsetHeight;
62     }
63     fs.style.left=left+"px";                                //浮动层位置改变
64     fs.style.top=top+'px';
65     var percentX=left/c.offsetWidth;                      //比例计算
66     var percentY=top/c.offsetHeight;
67     bp.style.left=-percentX*(bp.offsetWidth)+"px";        //右边大图位置的改变
68     bp.style.top=-percentY*(bp.offsetHeight)+"px";
69 }
70 }

```

2.11 图片的瀑布流

瀑布流效果是一种当下非常流行的图片展示布局，一般是一种多列，每一列中元素的宽度相等、高度不同，给人一种参差不齐却整体不乱的感觉。图 2.26 为这种布局的效果图。瀑布流效果是一种很好的图片展示方式，百度图片、美丽说、蘑菇街等许多网站都采用这种布局效果。在实际的应用中，往往是通过 AJAX 请求数据，达到动态添加的效果。

瀑布流在 CSS 布局方面比较简单，一般使用浮动定位和绝对定位都可以实现，只是在动态加载方面用 JavaScript 处理比较麻烦。可以说整个瀑布流效果的核心并不在 CSS 上，而 JavaScript 才是它的核心。



图 2.26 瀑布流

2.11.1 浮动的瀑布流

使用浮动制作瀑布流的显著优点是 CSS 代码较为简单，并且容易理解。这种方式的特点是 CSS 定位容易，但 JavaScript 动态加载较复杂。HTML 结构和 CSS 代码如下：

```
//HTML 部分
01 <div class="pubuliu">
02 <ul>
03   <li><p>标题</p></li>
04   更多的 li
05 </ul>
06 更多的 ul
07 </div>

//CSS 部分
01 *{margin:0;padding:0;}
02 .pubuliu{
03   width:708px;
04   height:auto;           /*居中*/
05   margin:20px auto;
06 }
07 .pubuliu>ul{
08   width:226px;
09   margin:5px;
10   float:left;
11   list-style:none;
12 }
```

在 HTML 中每一个 ul 对应网页中每一列图片，一般在瀑布流布局中网页上的每一列图片都会使用相同的宽度，在 CSS 中第 8 行每个 ul 的宽度是与图片相同的。第 9 行每个 ul

留 5 像素的外边局使它们保持距离。第 10 行使 ul 左浮动。在第 3 行，整体的宽度需要依据每列宽度和外边局计算，使所有的列都能盛放在 div 中。并使用第 4 行使其居中显示。这样瀑布流的布局就完成了。

无论是浮动定位还是绝对定位，要实现动态加载，在 JavaScript 方面所需要做的是基本一致的，实现的过程中有以下几个主要步骤：

- 确定如何在使用 JavaScript 之前手动地修改 HTML 代码模拟动态修改
- 如果使用 JavaScript，需要在什么情况下触发执行
- 如何实现触发
- 触发之后 JavaScript 需要做什么工作
- 写 JavaScript 代码与修复错误

基于上面的思路，考虑实际的问题：

- HTML 中每一列 ul 的每一个 li 元素为每块图片的内容，当增减 ul 中的 li 的数量时，体现在网页上便是每一列图片的增减，所以 JavaScript 需要做的是修改这些 li 来代替手动添加的过程。
- 瀑布流的期望是当滚动条拖动到页面的最后一幅图片（不管它在哪一列）时，加载新的图片和内容，然后使这些内容变为每个 li 元素插入到对应的列中。
- 如何判断滚动条是否已经拖动到了底部？首先获取到窗口的可用高度范围 d1，在每一次滚动条滚动时，获取到滚动条滚动过的距离 d2。遍历所有的列，取得每一列最后一个元素的高度 d3 以及该元素距离文档顶部的距离 d4。当 $d3+d4 < d1+d2$ 时，大致（因为取得的值没有取整，所以为大致，小数的影响可以忽略）可以判定滚动到了每一列的最后一个元素底部，这时便需要加载新的数据。在实际应用中，往往通过 AJAX 请求一个 url。

注意：为了便于测试，本例的数据为固定数据，所有的数据直接以 JSON 的方式写在了 JavaScript 文件中，AJAX 请求这一文件进行操作，另外本例是无限加载，实际中有限加载时使用布尔变量做一个开关控制与动态 URL 结合即可。

- 触发之后需要执行的必然是 AJAX 请求，请求到数据之后，把数据组成一串 HTML 代码，添加到相应的列中即可。
- 最后写 JavaScript 代码与修复错误

注意：在写 JavaScript 时常常遇到错误，建议使用浏览器的控制台进行调试，Chrome 浏览器、Opera 浏览器和 Firefox 浏览器均有调试台的功能，能够指明错误的地点，并且可以在 JavaScript 通过 `console.log()` 在控制台做数据记录，是很好的工具。

下面的这段代码是浮动定位瀑布流实例中基于 jQuery 的 JavaScript 代码，在当前主流浏览器上测试均可正常工作。代码中做了详细的注释，结合上述的分析可快速理解。

```
01 <script type="text/javascript" src="js/jq.js"></script>
02 <script type="text/javascript">
03 $(function (){
04   var veiw_height=$(window).height();           //获取窗口可以使用范围的高度
```

```

05  var str;                                     //HTML 组建时的字符串存放变量
06  $(window).resize(function(){
07      veiw_height=$(window).height();         //窗口大小改变时刷新可用范围的高度
08  });
09  $(window).scroll(function (){               //滚动条滚动触发
10      var uls=$(".pubuliu").children("ul");   //获取所有的列
11      for (var i=0;i< uls.length;i++ )        //遍历列
12      {
13          var lastli=uls.eq(i).children("li").last(); //获取每一列的最后一个元素
14          var top_distance =lastli.offset().top;    //获取最后一个元素距离文档顶部的距离
15          var listli_height=parseInt (lastli.css("height")); //获取最后一个元素的高度值
16          var scroll_distance=document.documentElement.scrollTop|| document.body.scrollTop;
17                                                    //获取滚动条向下滚动的距离
18          if(top_distance+listli_height<veiw_height+scroll_distance) //判断是否到达了底部
19          {
20              ajax(i);                             //需要加载新内容，ajax 请求
21          }
22      }
23  });                                           //滚动结束
24  });                                           //主函数结束
25  function ajax(number){
26      $.ajax({url:"js/data.js ",
27          success:function(data)                 //回调函数
28          {
29              str="";                             //清空字符串变量
30              data=eval(data);                     //解析 json
31              var n2=data[number].src.length;
32              for (var j=0;j<n2;j++){              //遍历数据
33                  str+="

```

2.11.2 绝对定位的瀑布流

与浮动瀑布流不同，绝对定位的瀑布流使用唯一的 ul，所有的小块元素都包含在 ul 中。在 CSS 定位较麻烦，需使用 JavaScript 辅助定位，但在动态加载新数据方面较为简单，只需把获取到的数据构建为 HTML 代码然后添加到唯一的一个 ul 下。绝对定位的 CSS 如下：

```

01  *{margin:0;padding:0;}
02  .pubuliu{

```

```

03     position:relative;
04     width:708px;
05     height:auto;                /*居中*/
06     margin:20px auto;
07 }
08 .pubuliu>ul>li{
09     list-style:none;
10     position:absolute;
11 }

```

与浮动定位的不同之处在于第 8~11 行，直接使 li 元素的 position 为 absolute，为了使之能够相对于父元素定位，第 3 行将整体 position 设为 relative。在绝对定位下，需要判断每个元素的 left 值和 top 值。因为图片的宽度是一致的，所以对 left 的计算方法可以采用列数求值法，即用当前的 li 的索引（li 为所有元素中的第几个）模列数（本例为 3 列）取得当前元素所在的列，然后计算出 left 值。对于 top 的计算方法则采用行累积法，即元素的 top 值为该列中其上所有元素的高度和空隙的累加值。因为各元素的高度不尽相同，所以在计算前需在遍历过程中记录每个元素的高度值并存入数组，在累加时采用 li 的索引除以列数（本例为 3 列）向下取整之后的值为循环上限做累加，具体情况可参照以下的 jQuery 代码。

```

01 <script type="text/javascript" src="js/jq.js "></script>
02 <script type="text/javascript">
03 $(function (){
04     var lis=$(".pubuliu").children("ul").children("li");    //获取所有 li 元素
05     var li_height = {C1:[],C2:[],C3:[]};                  //元素高度暂存数组
06     for (var i=0;i<lis.length;i++)
07     {
08         var col=i%3;                                        //计算出元素应在的列
09         switch(col)                                       //分情况
10         {
11             case 0:                                        //第一列
12                 lis.eq(i).css("left","5px");
13                 li_height.C1.push(parseInt (lis.eq(i).css("height")));
14                 //把当前元素的高度存入数组 C1
15                 var row=Math.floor(i/3);                 //向下取整求排数
16                 if(!row)                                   //第一排
17                 {
18                     lis.eq(i).css("top","0");
19                 }else                                     //非第一排
20                 {
21                     var top=0;                             //top 值暂存器
22                     for (var j=0;j<row ;j++ )
23                     {
24                         top+=li_height.C1[j];              //累加元素上方所有排元素的高度
25                     }

```

```

26         lis.eq(i).css("top",top+"px");
27     }
28     break;
29     case 1:                                     //第 2 列
30         lis.eq(i).css("left", "236px");
31         li_height.C2.push(parseInt (lis.eq(i).css("height")));
32                                             //把当前元素的高度存入数组 C2
33         var row=Math.floor(i/3);
34         if(!row)
35         {
36             lis.eq(i).css("top", "0");
37         }else
38         {
39             var top=0;
40             for (var j=0;j<row ;j++ )
41             {
42                 top+=li_height.C2[j];
43             }
44             lis.eq(i).css("top",top+"px");
45         }
46         break;
47     case 2:
48         lis.eq(i).css("left", "472px");
49         li_height.C3.push(parseInt (lis.eq(i).css("height")));
50                                             //把当前元素的高度存入数组 C3
51         var row=Math.floor(i/3);
52         if(!row)
53         {
54             lis.eq(i).css("top", "0");
55         }else
56         {
57             var top=0;
58             for (var j=0;j<row ;j++ )
59             {
60                 top+=li_height.C3[j];
61             }
62             lis.eq(i).css("top",top+"px");
63         }
64         break;
65     }
66 }
67 })
68 </script>

```

判断是否需要加载及 AJAX 请求的方法与浮动定位相似并且添加时更简单，只需把组

建的 HTML 代码直接添加到 ul 中即可，无须判断元素是在哪一列中，因为元素的定位是由上面的 JavaScript 承担的。这里不再赘述。

注意：本节所讲的瀑布流实现重点在实现原理，对于美化方面可以自由发挥，具体样式可以参考名站的做法。

2.12 图片墙

图片墙是一种用来展示一系列图片的网页，用户可以使用鼠标拖动图片查看。本节结合本章中拍立得效果框、CSS 3D 转换和拖曳结合起来制作的图片墙效果如图 2.27 所示。



图 2.27 图片墙

在 HTML 部分每一张图片的结构都是下面这样：

```
01 <div class="polaroid">
02 <div class="cover"></div>
03 
04 <p>图片说明文字</p>
05 </div>
```

最外层的 div 设置拍立得效果，第 2 行中的 div 为最外层的遮罩层，目的是避免图片文字被选中或被拖曳。CSS 部分则需要以下的设置，其中不包括 2D 转换中的 transform 属性，该属性由 JavaScript 部分随机生成。

```
01 *{margin:0;padding:0;}
02 body{background:url("img/bg.jpg");overflow:hidden;}
03 .polaroid{padding:10px;background:#FFF;text-align:center;width:auto;position:absolute;}
04 .cover{position:absolute;width:100%;height:100%;z-index:10;}
```

拍立得效果框由 padding 生成（与曾经的实例不同，是另一种实现方式），当然也可以使用图片作为背景来生成，对图片设置绝对定位，在 JavaScript 部分改变 left 和 top 实现拖曳。第 4 行中的遮罩层保持在图片和文字的上方。

JavaScript 部分需要完成的工作有随机产生定位值和旋转度数、当单击某张图片时使该

图片调整到最上层、随意拖动图片和使用键盘的左右箭头对图片的旋转度数进行调整。下面是一种可行的 **JavaScript** 方案，代码较为简洁。但在获取当前的旋转度数、获取窗口大小和实现拖动等次要方面全部写在一个 **js** 文件中，以下只说明与 **CSS** 相关的主要方面。

```

01 <script type="text/javascript" src='js/zQuery.js'></script> //次要部分函数
02 <script type="text/javascript">
03 window.onload =function (){
04     var ocover=getbyclassname("div","cover"); //获取遮罩层
05     var pre=ocover[0].parentNode; //在图片切换时存放上一张图
06     var body=document.getElementsByTagName('body')[0]; //获取 body 供窗口大小的设置
07     resize(); //随机参数生成
08     window.onresize=resize; //页面大小调整刷新参数
09     function resize(){ //随机图片布局的生成
10         var winh=win('height');var winw=win('width'); //获取窗口大小
11         body.style.width=winw; //设置 body 宽度
12         body.style.height=winh; //设置 body 高度
13         for (var i=0;i<ocover.length ;i++ )
14         {
15             var left=Math.random()*(winw-1.3*ocover[i].offsetWidth); //随机 left
16             var top=Math.random()*(winh-ocover[i].offsetHeight); //随机 top
17             var deg=Math.random()*45; //随机角度
18             if (i%2==0)
19             {
20                 deg=-deg; //图片顺、逆时针旋转交替进行
21             }
22             ocover[i].parentNode.style.left=left+'px'; //设置定位值
23             ocover[i].parentNode.style.top=top+'px';
24             ocover[i].parentNode.style.webkitTransform='rotate('+deg+"deg)"; //设置旋转度数
25             ocover[i].parentNode.style.mozTransform='rotate('+deg+"deg)";
26             ocover[i].parentNode.style.oTransform='rotate('+deg+"deg)";
27             ocover[i].parentNode.style.msTransform='rotate('+deg+"deg)";
28             ocover[i].parentNode.style.transform='rotate('+deg+"deg)";
29             ocover[i].onmousedown=function (){ //选中图片
30                 pre.style.zIndex=0; //上一张图片的 zIndex 变为 0
31                 this.parentNode.style.zIndex=1000; //当前图片调整到最上层
32                 drag(this.parentNode); //实现当前图片的拖曳
33                 pre=this.parentNode; //更新上一张图片为本图
34             }
35         }
36     }
37     document.onkeydown=function(event){ //键盘按下
38         var deg=css(pre,'rotate')[0]; //获取图片当前的旋转度数
39         if(event.keyCode==37){ //判断按键（左箭头）
40             deg--; //度数减小
41             if (deg<-90)

```



```

42     {
43         deg++;                                //防止颤抖(考虑到方法 css())
44     }
45 }else if(event.keyCode==39){                //判断按键(右箭头)
46     deg++;
47     if (deg>90)
48     {
49         deg--;
50     }
51 }else{
52     return false;
53 }
54 pre.style.webkitTransform='rotate('+deg+"deg)";    //设置角度
55 pre.style.mozTransform='rotate('+deg+"deg)";
56 pre.style.oTransform='rotate('+deg+"deg)";
57 pre.style.msTransform='rotate('+deg+"deg)";
58 pre.style.transform='rotate('+deg+"deg)";
59 }
60 }
61 </script>

```

注意：使用 JavaScript 设置 rotate 属性的值虽然简单，但获取当前的旋转度数区会返回 matrix 方法的六个参数，在处理时要根据得到的值进行处理得到当前的度数。由于篇幅原因，次要部分的实现可参见另一 js 代码。

2.13 图片轮播图

图片轮播图，也被称为焦点图，是网页设计中使用最多的效果之一，多数网站上都能看到图片轮播图的身影。曾经有一项调查显示，图片轮播图的转化率能达到很高的水平，很多网站把焦点内容以这种形式展现出来，于是图片轮播图又名焦点图。图片轮播图样式多种多样，JavaScript 的实现也不尽相同，但实现的原理有限，常用的有定位法、显示隐藏法和无缝切换等。本节就常用的几种方法分别实现同一款图片轮播图的不同效果，它们的长相相同，只是切换的方式不同，图 2.28 为效果图。

图片轮播图的基本功能有点击箭头切换到上一页/下一页、点击下方的小圆点显示对应的图片、自动播放以及鼠标悬浮在图片上方时显示箭头、按钮、暂停播放的功能。

HTML 部分的结构如下，结构并不是死板的，可以根据需求修改。

```

01 <div id="pics">
02 <ul class="pics">
03 <li><a href=""></a></li>
04 <li><a href=""></a></li>
05 <li><a href=""></a></li>

```

```

06 <li><a href=""></a></li>
07 <li><a href=""></a></li>
08 </ul>
09 <span class="pics_pre"></span><span class="pics_next"></span>
10 <ul class="pics_list"></ul>
11 </div>

```

第 1 行为外层容器，是为了方便元素获取而设置，第 2~8 行为图片列表，第 9 行为上一页和下一页按钮，第 10 行为图片下方的圆点列表，在 JavaScript 部分会根据图片的数量动态地加入 li 子元素。



图 2.28 图片轮播图

2.13.1 使用定位实现

使用定位实现即外层容器设为溢出隐藏，容器内的 ul 整体做相对定位，通过定位属性值改变可以看到的部分。通常可以使用 left 属性制作左右切换的效果，使用 top 属性制作上下切换的效果。实现原理如图 2.29 所示。

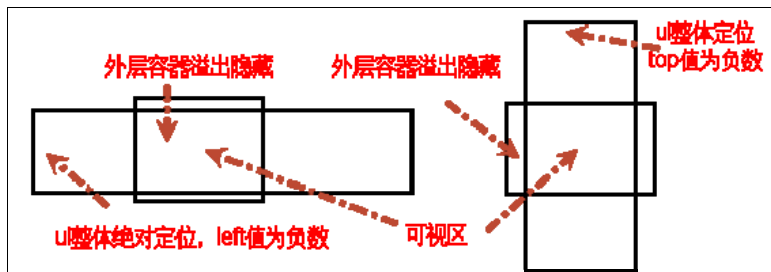


图 2.29 定位轮播图原理

在使用 left 实现左右切换时，CSS 部分如下：

```

01 *{margin:0;padding:0;}
02 #pics{ /* 容器 */

```

```

03     width:520px;height:280px;
04     border:1px solid;overflow:hidden;
05     position:absolute;top:100px;left:300px;
06 }
07 .pics{
08     width:2600px;height:280px;
09     position:relative;
10     left:0;
11 }
12 .pics>li{float:left;}           /*左浮动排列在一起*/
13 .pics_pre{
14     width:32px;height:32px;
15     position:absolute;top:45%;left:0;
16     background:url("img/arrow_left.png"); /*左箭头*/
17     cursor:pointer;display:none;
18 }
19 .pics_next{
20     width:32px;height:32px;
21     position:absolute;top:45%;right:0;
22     background:url("img/arrow_right.png"); /*右箭头*/
23     cursor:pointer;display:none;
24 }
25 .pics_list{                     /*小圆点区*/
26     width:100%;height:8%;
27     position:absolute;bottom:0;
28     background:#8B8878;opacity:0.8;filter:alpha(opacity:80);
29     cursor:pointer;text-align:center;display:none;
30 }
31 .pics_list>li{                 /*圆点*/
32     width:10px;height:10px;
33     border-radius:5px;
34     background:#ffffff;
35     cursor:pointer;
36     float:left;margin:5px;left:35%;position:relative;
37     list-style:none;
38 }

```

需要注意的是第4行的溢出隐藏、第8行的ul整体宽度设置（只有设置为大数值，图片才能显示在一排中）、第9~10行的整体定位（JavaScript部分改变left值以及第12行的左浮动（JavaScript部分需要获取每幅图的相对于ul的偏移值供left定位使用）。其余的按钮和小圆点部分多采用绝对定位的方式布局到合理位置。

在JavaScript部分，需要完成基本的功能，这段JavaScript代码可以实现功能，元素运动部分的实现是通过运动框架中的move函数实现，其作用与jQuery中的animate函数相似。

```

01 <script type="text/javascript" src="js/zQuery.js"></script> //加载运动框架
02 <script type="text/javascript">
03 window.onload=function(){
04     var pics=document.getElementById('pics');           //获取外层 div
05     var pics_pre=getbyclass(pics,'pics_pre')[0];        //获取上一个按钮
06     var pics_next=getbyclass(pics,'pics_next')[0];      //获取下一个按钮
07     var pics_list=getbyclass(pics,'pics_list')[0];      //获取圆点 ul
08     var pics_ul=pics.getElementsByTagName('ul')[0];     //获取图片 ul
09     var pics_lis=pics_ul.getElementsByTagName('li');    //获取图片 li
10     var inow=0;                                          //贯穿整体的当前图片索引变量
11     for (var i=0;i<pics_lis.length;i++ )
12     {
13         var list=document.createElement('li');
14         pics_list.appendChild(list);                    //为圆点 ul 添加所有的小圆点
15     }
16     var list_li=pics_list.getElementsByTagName('li');    //获取所有小圆点
17     for (var i=0; i<list_li.length;i++ )
18     {
19         list_li[i].onclick=function (){                //为每个小圆点加载点击事件
20             inow=index(this,list_li);                  //获取当前圆点在圆点 ul 中的索引值
21             show(inow);                                 //展示相应的图片
22         }
23     }
24     show(0);                                             //页面加载完成显示第一张
25     var timer=setInterval(function (){                 //定时器自动播放
26         if (inow<pics_lis.length-1)
27         {
28             inow++;                                     //页面加载完成后将从第二张开始，解决定时器等待的问题
29         }else{
30             inow=0;
31         }
32         show(inow);
33     },3000);
34     pics_pre.onclick=function (){                      //上一个按钮被点击
35     if (inow>0)                                         //更改索引变量的值
36     {
37         inow-=1;
38     }else{
39         inow=pics_lis.length-1;
40     }
41     show(inow);                                         //展示
42 }
43     pics_next.onclick=function (){                    //下一个按钮被单击
44     if (inow<pics_lis.length-1)                       //更改索引变量的值
45     {

```

```

46     inow+=1;
47 }else{
48     inow=0;
49 }
50     show(inow);                //展示
51 }
52     function show(inow){        //展示函数
53         var l=pics_lis[inow].offsetLeft;    //获取到需要展示的图片的偏移量
54         move(pics_ul,{left:-l});            //偏移量作为运动中 left 值
55         for (var i=0;i<pics_lis.length ;i++ )
56         {
57             list_li[i].style.background='#FFFFFF';    //将所有小圆点的背景色变为白色
58         }
59         list_li[inow].style.background='#EE7600';    //将当前小圆点的颜色改变
60     }
61     pics.onmouseover=function(){        //鼠标悬停在图片上方
62         pics_pre.style.display="block";    //展示按钮与小圆点
63         pics_next.style.display="block";
64         pics_list.style.display='block';
65         clearInterval(timer);            //暂停播放
66     }
67     pics.onmouseout=function(){        //鼠标移开图片
68         pics_pre.style.display="none";    //将按钮与小圆点隐藏
69         pics_next.style.display="none";
70         pics_list.style.display="none";
71         timer=setInterval(function (){        //重开定时器，展示第 inow 张图片
72             if (inow<pics_lis.length)
73             {
74                 show(inow);
75                 inow++;
76             }else{
77                 inow=0;
78             }
79             },3000);
80     }
81 }
82 </script>

```

而使用 top 值实现上下切换时，仅需在以上的代码中稍作改变即可。在 CSS 中去掉图片 ul 的大小设置，定位的 top 值设为 0，同时在 JavaScript 中将 offsetLeft 改为 offsetTop、left 改为 top 即可，部分修改如下所示。

//CSS 代码

```
01 .pics{position:relative;top:0;}
```

//JavaScript 代码

```

01 function show(inow){
02   var l=pics_lis[inow].offsetTop;
03   move(pics_ul,{top:-l});

```

2.13.2 使用透明度实现

在代码上，使用透明度实现与使用定位实现也相似，稍稍修改代码即可。与定位不同的是使用透明度实现的是淡入淡出的效果。它的原理是 CSS 中对所有的 li 绝对定位使其脱离文档流重叠在一起，并全部设置为透明，通过改变 CSS 透明度属性使对应的图片显示。在显示图片前，遍历所有的图片并全部隐藏，然后再让需要显示的图片显示出来。与定位不同的代码如下：

```

//CSS 代码
01 .pics>li{position:absolute;opacity:0;filter:alpha(opcity:0);}
//JavaScript 代码
01 function show(inow){                                     //展示函数
02   for (var i=0;i<pics_lis.length ;i++ )
03   {
04     move(pics_lis[i],{opacity:0});
05     list_li[i].style.background='#FFFFFF';
06   }
07   move(pics_lis[inow],{opacity:100});
08   list_li[inow].style.background='#EE7600';
09 }

```

2.13.3 无缝切换

无缝切换效果是对定位效果的优化，在定位效果中，在第一张图片与最后一张图片切换时，由于 left 值的变化很大，导致元素的运动幅度很大，给用户一种晃眼的感觉。无缝切换效果解决的就是这一问题，在无缝切换中，第一张图片与最后一张图片切换时移动的仅仅是一张图片的距离，而不是定位切换中移动多张图片的距离。由于是定位效果的优化，所以在原理上二者有些相似，原理图如图 2.30 所示。

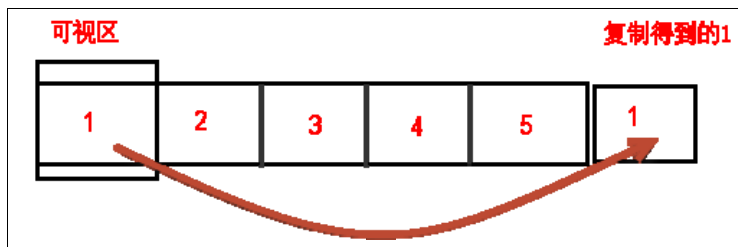


图 2.30 无缝切换原理图 1

以图中切换到下一张图片为例，当单击下一张按钮时，首先将位于列表最左端的图片 1 复制，添加在 ul 的最右端，然后使整个 ul 向左运动一张图片的距离，这时图片 2 将位于

可视区中，左边的图 1 位于可视区的左边，如图 2.31 所示。

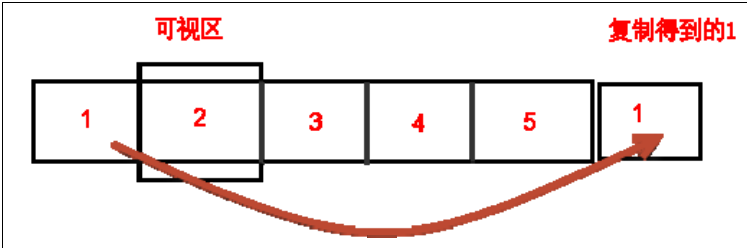


图 2.31 无缝切换原理图 2

这时把最左边的图片 1 删除，会得到图 2.32 所示的效果，这并不是期望的结果，需要重新将 ul 的 left 值设为 0 以达到图 2.33 所示的效果。

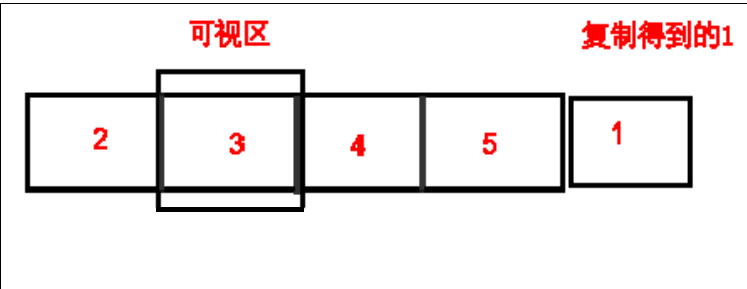


图 2.32 无缝切换原理图 3

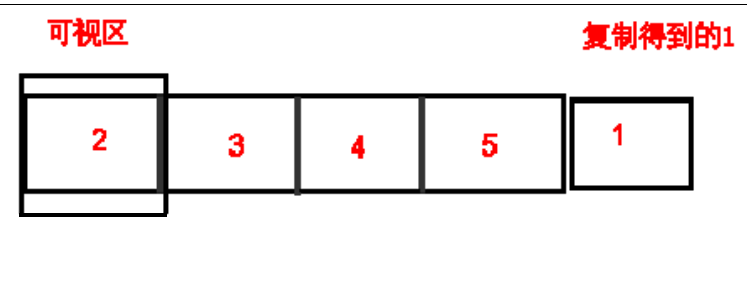


图 2.33 无缝切换原理图 4

再次切换下一张时同理，显然图片 5 和图片 1 是紧挨着的，于是切换时仅运动一张图片的距离。

与定位轮播图相比，在实例中，仅需加宽 ul 的宽度（事实上 ul 的宽度应该由 JavaScript 部分动态计算得到）。JavaScript 部分改动较多，一种解决方案如下，代码中保留了 show 方法以便在点击小圆点时仍然可以切换，添加了展示上一张和展示下一张的函数。

```
01 <script type="text/javascript" src="js/jquery.js"></script>
02 <script type="text/javascript">
03 window.onload=function(){
04   var pics=document.getElementById('pics');           //获取外层 div
05   var pics_pre=getbyclass(pics,'pics_pre')[0];       //获取上一个按钮
06   var pics_next=getbyclass(pics,'pics_next')[0];     //获取下一个按钮
```

```

07 var pics_list=getbyclass(pics,'pics_list')[0];           //获取圆点 ul
08 var pics_ul=pics.getElementsByTagName('ul')[0];         //获取图片 ul
09 var pics_lis=pics_ul.getElementsByTagName('li');         //获取图片 li
10 var n=pics_lis.length;                                   //图片数量
11 var inow=0;                                              //当前图片索引变量
12 var btn=true;                                           //开关，解决连续点击时删除元素的问题
13 for (var i=0;i<pics_lis.length ;i++ )
14 {
15     var list=document.createElement('li');
16     pics_list.appendChild(list);                         //为圆点 ul 添加所有的小圆点
17 }
18 var list_li=pics_list.getElementsByTagName('li');       //获取所有小圆点
19 for (var i=0; i<list_li.length;i++ )
20 {
21     list_li[i].onclick=function (){                     //为每个小圆点加载点击事件
22         inow=index(this,list_li);                       //获取当前圆点的索引值
23         show(inow);                                     //展示相应的图片
24     }
25 }
26 show(0);                                                 //页面加载完成显示第一张
27 var timer=setInterval(function (){                     //定时器自动播放
28     if (inow<n-1)
29     {
30         inow++;                                         //页面加载完成后将从第二张开始，解决定时器等待的问题
31     }else{
32         inow=0;
33     }
34     show_next();                                       //展示下一张
35 },3000);
36 pics_pre.onclick=function (){                           //上一个按钮被点击
37     show_pre();                                       //展示上一张
38 }
39 pics_next.onclick=function (){                          //下一个按钮被点击
40     show_next();                                       //展示下一张
41 }
42 function show_pre(){                                    //展示上一张函数
43     if (btn)
44     {
45         btn=false;                                     //将开关关闭，直到下次打开之前点击按钮将无效
46         newli=pics_lis[n-1].cloneNode(true);          //复制最后一张
47         pics_ul.insertBefore(newli,pics_lis[0]);      //添加到第一张之前成为第一张
48         pics_ul.style.left=-pics_lis[1].offsetLeft+'px';//拉到第二张的位置
49         move(pics_ul,{left:0},'buffer',function(){    //从第二张图切换到第一张图

```



```

50         pics_ul.removeChild(pics_lis[n]);    //移除最后一张图
51         btn=true;                            //删除完毕后打开开关
52     });
53     if(inow>0){                               //更改索引值
54         inow--;
55     }else{
56         inow=4;
57     }
58     for (var i=0;i<n ;i++ )
59     {
60         list_li[i].style.background='#FFFFFF';//将所有小圆点的背景色变为白色
61     }
62     list_li[inow].style.background='#EE7600'; //将对应小圆点的颜色改变
63 }
64 }
65 function show_next(){
66     if (btn)
67     {
68         btn=false;                            //将开关关闭，直到下次打开之前点击按钮将无效
69         var newli=pics_lis[0].cloneNode(true); //复制第一张
70         pics_ul.appendChild(newli);           //添加到 ul 最后
71         var l=pics_lis[1].offsetWidth;
72         move(pics_ul,{left:-l},'buffer',function(){ //切换至第二张图
73             pics_ul.removeChild(pics_lis[0]);    //移除第一张图
74             pics_ul.style.left=0;                //重设 left 为 0
75             btn=true;                            //打开开关
76         });
77         if(inow<n-1){                           //更改索引值
78             inow++;
79         }else{
80             inow=0;
81         }
82         for (var i=0;i<n ;i++ )
83         {
84             list_li[i].style.background='#FFFFFF';//将所有小圆点的背景色变为白色
85         }
86         list_li[inow].style.background='#EE7600'; //将对应小圆点的颜色改变
87     }
88 }
89 function show(inow){                          //展示函数
90     var l=pics_lis[inow].offsetLeft;          //获取到需要展示的图片的偏移量
91     for (var i=0;i<n ;i++ )
92     {
93         list_li[i].style.background='#FFFFFF'; //将所有小圆点的背景色变为白色

```

```

94  }
95  list_li[inow].style.background='#EE7600';           //将当前小圆点的颜色改变
96  move(pics_ul,{left:-l},'buffer');                 //偏移量作为运动中 left 值
97  }
98  pics.onmouseover=function(){                      //鼠标悬停在图片上方
99  pics_pre.style.display='block';                  //展示按钮与小圆点
100  pics_next.style.display="block";
101  pics_list.style.display='block';
102  clearInterval(timer);                             //暂停播放
103  }
104  pics.onmouseout=function(){                       //鼠标移开图片
105  pics_pre.style.display="none";                   //将按钮与小圆点隐藏
106  pics_next.style.display="none";
107  pics_list.style.display="none";
108  timer=setInterval(function (){                    //重开定时器
109      if (inow<pics_lis.length)
110      {
111          show_next();
112          inow++;
113      }else{
114          inow=0;
115      }
116  },3000);
117  }
118  }
119  </script>

```

2.14 幻灯片

在这一节中，将介绍一种图片展现方式——幻灯片效果。幻灯片效果与焦点图效果有相同之处，通常来说，图片滚动的原理是相同的。本节实例的原型为腾讯视频首页的幻灯片效果，代码上采用 CSS 布局+jQuery 实现，效果如图 2.34 所示。



图 2.34 幻灯片

在效果上，大图为核心，并采用淡入淡出效果展示。大图的下部分有视频的文字说明，前一个和后一个按钮，以及可以横向滚动的小图集合。整体可以自动播放，鼠标在幻灯片上悬浮时可以暂停播放。HTML 部分的代码结构如下：

```
01 <div class="powerpoint">
02 <div class="powerpoint_big_pic">
03 大图 ul, ul 中有多个 li
04 </div>
05 <div class="powerpoint_textlist">
06 文字说明 ul, ul 中有多个 li
07 </div>
08 <div class="powerpoint_pre"><</div>
09 <div class="powerpoint_small_pic">
10 小图 ul, ul 中有多个 li
11 </div>
12 <div class="powerpoint_next">></div>
13 </div>
```

CSS 布局部分主要工作为定位各部分的位置。大图的 li 和文本说明的 li 使用绝对定位使所有元素重合，小图 ul 则需使用相对定位。

```
01 *{margin:0;padding:0;list-style:none;}
02 .powerpoint{
03     position:absolute;
04     width:100%; height:460px;
05     background:black;
06     overflow:hidden;
07 }
08 .powerpoint_big_pic>ul>li{
09     position:absolute;top:0;left:-200px; /*图偏宽度大于屏幕宽度，大图居中*/
10     width:100%;height:100%;
11     opacity:0;
12 }
13 .powerpoint_textlist{
14     position:absolute;top:78%;left:10%;
15     width:20%;height:15%;
16     background:#7A7A7A;
17     font-size:20px;
18     color:#FFFFFF;
19 }
20 .powerpoint_textlist>ul>li{
21     position:absolute;
22     opacity:0; /*文字的原理同大图的原理*/
23 }
24 .powerpoint_small_pic{
25     position:relative;top:78%;left:35%;
26     width:51%;height:15%;
```

```

27     background:#7A7A7A;
28     overflow:hidden;
29 }
30 .powerpoint_small_pic>ul{
31     position:relative;
32     width:1500px;                /*设为大数以便 li 排成一排*/
33 }
34 .powerpoint_small_pic>ul>li{
35     margin:5px;
36     float:left;                /*左浮动*/
37 }
38 .powerpoint_pre,.powerpoint_next{
39     position:absolute;top:81%;
40     width:37px;height:38px;
41     background:#303030;
42     font-size:40px;
43     color:#FFFFFF;
44     border-radius:20px;
45     cursor:pointer;
46 }
47 .powerpoint_pre{left:31%;}
48 .powerpoint_next{left:87%;}

```

应用上面的 CSS 代码即可得布局效果，通过手动在第 31 行上添加 `left` 值，即可改变小图整体的位置，从而达到小图滚动的效果，这与定位法制作轮播图的原理相同。而在第 11 行和第 22 行，所有的元素都是透明的，当其中某个元素不透明时，反映在效果上就是图片展现。

在 JavaScript 部分，需要实现以下几点：

(1) 单击小图，对应的小图上出现橙色边框，同时当前大图淡出，对应的大图淡入，对应文字说明同理。

(2) 单击按钮，小图切换（小图切换时需要考虑 `left` 取值的限制），同时带动大图切换。

(3) 实现图片自动播放，鼠标悬停在幻灯片上暂停播放。

为了明确地表明 JavaScript 的实现过程，本实例采用 jQuery，因为淡入淡出、缓慢切换的效果都可由 jQuery 中的 `animate` 来实现。如果在某些场合下不允许使用 jQuery，那么使用原生 JavaScript 实现以上的功能即可。下面是一种可行的 jQuery 方案，每一行语句都有注释说明，主流浏览器均可通过。

```

01 <script type="text/javascript" src="js/jq.js"></script>
02 <script type="text/javascript">
03 $(function(){
04     var p=$(".powerpoint");                //获取整体，自动播放与暂停播放用到
05     var pspu=$(".powerpoint_small_pic>ul");    //获取小图 ul
06     var psp=$(".powerpoint_small_pic>ul>li");    //获取小图 li

```

```

07 var pbp=$(".powerpoint_big_pic>ul>li");           //获取大图 li
08 var ptl=$(".powerpoint_textlist>ul>li");          //获取文字说明 li
09 var pre=$(".powerpoint_pre");                     //获取上一个按钮
10 var next=$(".powerpoint_next");                   //获取下一个按钮
11 var now=0;                                         //记录当前被激活元素的变量,作为参数传入展示函数
12 showpic(0);                                       //页面载入完成播放第一张
13 var t=setInterval(function (){play()},4000);      //自动播放
14 p.hover(function (){clearInterval(t)},           //鼠标悬停清空定时器,暂停
15             function (){t=setInterval(function (){play()},4000)}); //鼠标移开继续播放
16 for (var i=0;i<psp.length ;i++ )                 //为每个小图写入点击事件
17 {
18     psp.eq(i).click(function (){
19         now=$(this).index();                       //将当前元素的索引存入变量
20         showpic(now);
21     });
22 }
23 pre.click(function(){                             //前一个按钮被点击
24     if(0==now){
25         now=psp.length;                             //范围限制
26     }
27     now=now-1;
28     showpic(now);
29 })
30 next.click(function(){                            //前一个按钮被点击
31     if(psp.length-1==now){
32         now=-1;                                       //范围限制
33     }
34     now=now+1;
35     showpic(now);
36 })
37 function play()                                    //自动播放函数
38 {
39     showpic(now);
40     now++;
41     if(pbp.length==now)
42     {
43         now=0;
44     }
45 }
46 function showpic(inow){                            //展示图片函数
47     for (var j=0;j<psp.length ;j++ )              //去除所有元素的可见性
48     {
49         pbp.eq(j).stop(true,false).animate({opacity:"0"},"normal");
50         ptl.eq(j).css("opacity","0");
51         psp.eq(j).css("border","0");

```

```

52  }
53      if(inow<6)                                //范围限制
54          left=psp.eq(inow).offset().left-psp.eq(0).offset().left;//计算小图标的左偏移值
55      else
56          left=psp.eq(6).offset().left-psp.eq(0).offset().left;//计算小图标的左偏移值
57          pbp.eq(inow).stop(true,false).animate({opacity:"1"},"normal");
58                                          //使编号为 now 元素可见
59          ptl.eq(inow).css("opacity","1");
60          psp.eq(inow).css("border","2px solid #FF7F00");
61          pspu.stop(true,false).animate({left:-left},"normal");//小图滚动（left 值改变）
62  }
63  });
64  </script>

```

2.15 手风琴效果

手风琴效果是图片的另一种展示方式，从效果上看，它与普通的图片轮播图有相似之处，JavaScript 部分的某些代码也是相似的。图 2.35 为效果图，标号 1~5 分别为不同的 5 张图片，在同一时期，5 张图片中只有 1 张图片可以完全显示出来，其余的 4 张图片为折叠状态。当鼠标不在图片上方，5 张图片每隔一段时间便会自动切换，鼠标位于图片上方时停止切换，鼠标单击某张图片时会展示被点击图片。



图 2.35 手风琴效果

与图片轮播图相比，手风琴的实现原理更加简单。在实例的 HTML 部分，5 张图片以列表呈现，每个列表项都左浮动排列在一起并且溢出隐藏，折叠的图片宽度较小，展示的图片宽度较大，列表自动撑开与左浮动的布局会使得所有的图片合理地排列在一起。纯 CSS 实现的手风琴效果如下：

```

01  *{margin:0;padding:0;}
02  li{
03      list-style:none;
04      float:left;

```

```

05     width:10px;                /*左浮动排列在一起*/
06     overflow:hidden;
07     border:1px solid;
08     transition:all .1s;        /*CSS 3 transition 动画效果*/
09     -webkit-transition:all .1s;
10     -moz-transition:all .1s;
11     -o-transition:all .1s;
12     cursor:pointer;           /*手型指针*/
13 }
14 .on{width:500px;}              /*第一张图默认显示*/
15 ul:hover li{width:10px;}      /*鼠标指向将所有的 li 恢复 10 像素*/
16 ul li:hover{width:500px;}     /*鼠标指向的当前 li 变为 500 像素*/

```

主要的部分是第 15 行和第 16 行，当鼠标指向某一项，首先使所有的 li 变窄，然后使当前指向的 li 变宽，顺序一定不能弄反，否则不会得到想要的效果。第 8~12 行加入了 CSS 3 过渡效果，使动作圆滑切换。

纯 CSS 代码实现功能存在局限性，不能自动播放，操作也同样受限。使用 JavaScript 则可以实现，实例中提供了一种 JavaScript 解决方案，使用 JavaScript 改变 li 的宽度，可自行查看实例。

2.16 图片自适应

图片自适应在响应式布局中颇有需求，所谓响应式布局，即为不同设备下布局的自行调整已达到适合设备屏幕的效果。本节以一个自适应的 logo 说明图片自适应的实现。如图 2.36 所示，使用浏览器模拟不同设备屏幕大小时，logo 图片可以自适应大小。

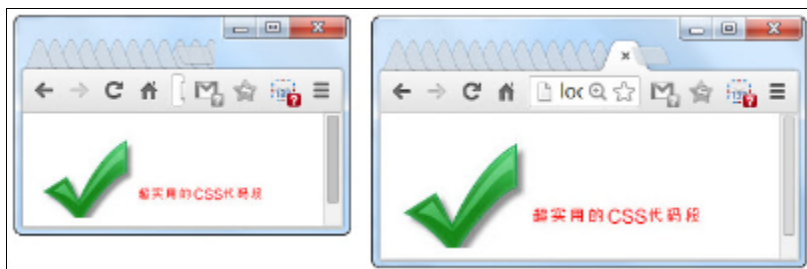


图 2.36 图片自适应

该效果在 HTML 部分使用的是 h1 标签，其样式为：

```

01 .logo {
02     background-image: url('img/logo.jpg');
03     background-size: 100%;
04     width: 100%;
05     padding-top: 29.8%;
06     height: 0;

```

```
07     text-indent: -9999px;
08 }
```

第 2 行中指定背景图的大小为 100%，第 3 行设置 h1 的宽度为浏览器当前宽度的 100%，第 5 行为 logo 上方留下一块空间，第 7 行把 h1 标签中的文字移动到屏幕可见区域之外（隐藏）。

2.17 使用纯 CSS 绘制图像

CSS 3 的属性为更复杂的图像绘制提供了方便，诸如使用 CSS 3 制作一个弯曲的元素、旋转一个元素等。本节的实例以一个纯 CSS 代码绘制的天猫 LOGO，讲述如何使用纯 CSS 代码绘制图像。图 2.37 为纯 CSS 绘制的天猫 LOGO 效果图。

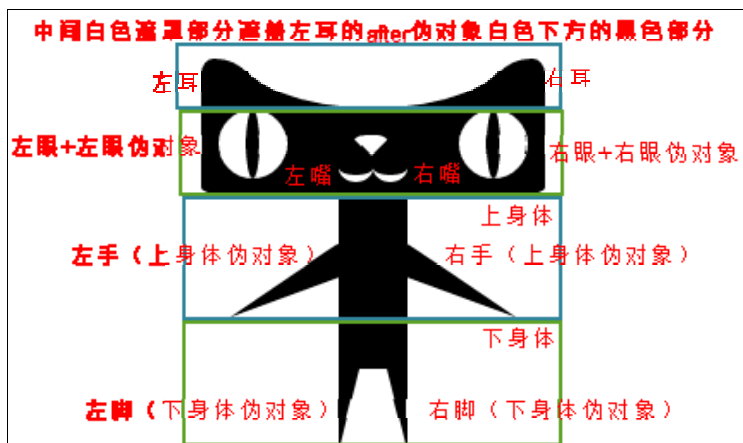


图 2.37 天猫 LOGO 效果图与分解

使用 CSS 绘制图像的关键在于 div 的变形、组合、定位以及伪对象的灵活使用。依据划分的结构在 HTML 部分建立可以设置大小的块级标签即可，本例中使用的是 div 和 span，HTML 5 中的 i 标签等也是可以的。在 CSS 部分对每个部分依次进行设置如下代码。

```
01 .ear{width:100%;height:15%;position:relative;} /*耳朵容器*/
02 .ear{ /*左耳*/
03     width:18%;height:100%;position:relative;display:inline-block;
04     border-top-left-radius:20% 30%; /*耳朵弧度圆角实现*/
05     border-top-right-radius:80% 100%;
06     background:black;
07     z-index:0;
08 }
09 .ear:after{ /*左耳朵的伪对象*/
10     content:"";width:450%;height:72%;
11     position:absolute;left:84%;bottom:0;
12     background:black; /*一块黑色的区域供耳朵中间容器遮盖*/
13 }
```



```

14 .earr{                                     /*右耳*/
15     width:18%;height:100%;
16     position:absolute;display:inline-block;right:0;
17     border-top-right-radius:20% 30%;         /*耳朵弧度圆角实现*/
18     border-top-left-radius:80% 100%;
19     background:black;
20     z-index:0;
21 }
22 .earm{                                     /*耳朵中间的遮盖层*/
23     width:76%;height:100%;
24     position:absolute;display:inline-block;left:12%;bottom:3%;
25     border-bottom-left-radius:50% 100%;       /*设置弧度*/
26     border-bottom-right-radius:50% 100%;     /*遮盖不到的地方为左耳伪对象的黑色部分*/
27     z-index:1;
28     background:#fff;
29     border-bottom:none;
30 }

```

因耳朵部分的实现需要圆角的组合实现弧线，耳朵部分为实例中最难实现的部分。左耳右耳使用圆角设置，中间使用另一个白色的圆角块遮盖住左耳的伪对象。通过调整使得圆弧与耳朵结合起来。

```

01 .face{width:100%;height:25%;              /*脸部*/
02     position:relative;background:black;
03     border-bottom-left-radius:10% 10%;       /*下方圆角*/
04     border-bottom-right-radius:10% 10%;
05 }
06 .eyel{                                     /*左眼白眼球*/
07     width:20%;height:80%;
08     position:absolute;left:5%;
09     border-radius:100%;                     /*圆形*/
10     background:white;
11     z-index:1;
12 }
13 .eyel:after{                             /*左眼黑眼球*/
14     content:"";
15     width:20%;height:100%;                 /*椭圆形*/
16     position:absolute;left:40%;
17     border-radius:100%;
18     background:black;
19     z-index:1;
20 }
21 .eyer{                                     /*右眼白眼球*/
22     width:20%;height:80%;
23     position:absolute;right:5%;
24     border-radius:100%;

```

```

25     background:white;
26     z-index:1;
27 }
28 .eyer:after{                                /*右眼黑眼球*/
29     content:"";
30     width:20%;height:100%;
31     position:absolute;left:40%;
32     border-radius:100%;
33     background:black;
34     z-index:1;
35 }

```

眼睛部分创建一个白色的圆形作为白眼球，然后使用 `after` 伪类创建一个黑色的椭圆作为黑眼球，剩余部分为定位。

```

01 .face:after{                                /*鼻子*/
02     content:"";
03     position:absolute;
04     border-top: 1em solid white;              /*使用边框*/
05     border-bottom: 1em solid transparent;
06     border-left: 1em solid transparent;
07     border-right: 1em solid transparent;
08     border-radius:38%;                       /*使用圆角*/
09     z-index:0;
10     left:46.5%;top:30%;
11 }

```

鼻子部分直接作为脸部的伪对象，利用边框创建三角形即可，使用圆角可以使三角形变得圆滑。

```

01 .mouthl{                                    /*嘴的左部分*/
02     width:10%;height:25%;
03     position:absolute;left:40%;top: 50%;
04     border-radius:100%;                      /*圆角*/
05     border-bottom: 0.4em solid white;        /*只用下边框*/
06     z-index:1;
07 }
08 .mouthr{                                    /*嘴的右部分*/
09     width:10%;height:25%;
10     position:absolute;left:50%;top: 50%;
11     border-radius:100%;
12     border-bottom: 0.4em solid white;
13     z-index:1;
14 }

```

嘴的实现直接使用一个圆角的下边框组合起来。而身体、手与腿部的实现使用带有伪对象的长方形实现，伪对象为使用边框创建的三角形，如下代码：

```

01 .bodytop{width:20%;height:25%;position:relative;left:40%;background:black;}/*身体上部分*/
02 .bodybottom{width:20%;height:25%;position:relative;left:40%;background:black;}/*身体下*/

```

```

03 .bodytop:before{                                     /*左手*/
04     position: absolute;right: 42%;top: 73%;
05     width: 50%;
06     content: "";
07     border-top: 0 solid transparent;                   /*边框创建三角形*/
08     border-bottom: 1.5em solid black;
09     border-left: 6em solid transparent;
10     border-right: 0 solid transparent;
11     -webkit-transform: rotate(-20deg);                 /*旋转三角形*/
12     -moz-transform: rotate(-20deg);
13     -o-transform: rotate(-20deg);
14     -ms-transform: rotate(-20deg);
15     transform: rotate(-20deg);
16 }
17 .bodytop:after{                                       /*右手*/
18     position: absolute;left:42%;top:73%;
19     width: 50%;
20     content: "";
21     border-top: 0 solid transparent;
22     border-bottom: 1.5em solid black;
23     border-left: 0 solid transparent;
24     border-right: 6em solid transparent;
25     -webkit-transform: rotate(20deg);
26     -moz-transform: rotate(20deg);
27     -o-transform: rotate(20deg);
28     -ms-transform: rotate(20deg);
29     transform: rotate(20deg);
30 }
31 .bodybottom:before{                                  /*左腿*/
32     position: absolute;left:0;top:100%;
33     width:50%;
34     content:"";
35     border-top: 0;                                     /*使用边框创建三角形*/
36     border-bottom: 4em solid transparent;
37     border-left: 1em solid black;
38     border-right: 0;
39 }
40 .bodybottom:after{                                   /*右腿*/
41     position: absolute;right:0;top:100%;
42     width:50%;
43     content:"";
44     border-top: 0;
45     border-bottom: 4em solid transparent;
46     border-right: 1em solid black;
47     border-left: 0;

```

48 }

看上去本节似乎不实用，因为天猫的图标是阿里巴巴公司的，其他网站用不上。实际上，本节的重点在于说明使用纯 CSS 代码描绘图形的方法。腾讯公司的 ISUX 官方网站上有一个腾讯企鹅图标的实现，而企鹅图标也是使用多块的组合来实现的。在 CSS 3 公布后不久，就有人专门研究使用 div 制作图标的方面，比较有名的是 One Div，在网页中使用图标仅需一个 div 和一套 CSS 样式即可替代图片。目前越来越多的网站使用图标文字或是单纯地 CSS 代码代替图片了，能够加快网站的响应。

2.18 图片原地放大

本节的实例效果图如图 2.38 所示，当鼠标移动到某一张图片上，该图片就会从中间原地放大显示。



图 2.38 图片原地放大效果图

初步看效果图可以知道，每张图片的定位方式都是绝对定位的，也就是 `absolute`。直接使用绝对定位必然导致所有的图片重叠在一起，而对每一张图片都制定 `left` 和 `top` 的方式不仅繁琐、效率低下，更重要的一点是，分别指定属性值的方式无法使任意数量的图片自适应，虽然 JavaScript 可以实现动态地赋值，但实现过程必定也会繁琐，灵活性差。实例中采用了一种快捷的方法解决这一问题，这种方法叫做布局转换。

布局转换首先在 CSS 部分使得所有的图片块左浮动在一起实现自适应，接着使用 JavaScript 遍历每张图片的 `offsetLeft` 和 `offsetWidth` 值并将其赋值给图片的 `left` 和 `width` 属性值，随后遍历每张图片并把图片设置为绝对定位（由于前一步已经设置好了定位值，这一步之后图片并不会堆叠在一起）。CSS 部分的代码如下：

```
01 .block{                                     /*图片外层容器*/
02     width:350px ;height:350px ;
03     border:1px solid;margin:150px auto;
04     position:relative;                       /*为图片的定位做准备*/
```

```

05  }
06  .b{                                     /* 图片*/
07      width:100px ;height:100px ;
08      border:1px solid;margin:5px;
09      float:left;                         /*左浮动自适应*/
10      list-style:none;overflow:hidden;
11  }
12  img{width:100px ;height:100px;}        /*图片适应容器尺寸*/

```

使用 JavaScript 实现布局转换，代码为：

```

01  window.onload =function (){
02  var block=document.getElementsByTagName('ul')[0]; //图片外层容器
03  var lis=block.getElementsByTagName('li'); //图片
04  for (var i=0;i<lis.length;i++){
05  lis[i].style.left=lis[i].offsetLeft+'px';
06  lis[i].style.top=lis[i].offsetTop+'px';
07  }
08  for (var i=0;i<lis.length;i++){
09  lis[i].style.position='absolute';
10  lis[i].style.margin=0;

```

为每张图片添加鼠标移入和鼠标移出事件，图片原地缩放的实现思路为尺寸值、左 margin 和上 margin 同时改变，这种方式比直接改变图片的 left 和 top 值更加快捷，因其不需要获取当前的 left 和 top 值，工作效率更高。

```

01  var izindex=1;                         //zindex 存放变量
02  for (var i=0;i<lis.length;i++){
03  lis[i].onmouseover=function (){
04      this.style.zIndex=izindex++;        //使得当前的图片保持最前端显示
05      var oimg=this.getElementsByTagName("img")[0]; //获取到 li 中的 img
06      move(this,{marginLeft: [-50],marginTop: [-50],width: [200],height: [200]}); //改变图片容器 li
07      move(oimg,{width: [200],height: [200]}); //改变图片，同 jQuery 中 animate
08  }
09  lis[i].onmouseout=function (){          //鼠标移开还原属性值
10      var oimg=this.getElementsByTagName("img")[0];
11      move(this,{marginLeft: [0],marginTop: [0],width: [100],height: [100]});
12      move(oimg,{width: [100],height: [100]});
13  }
14  }

```

注意：使用 CSS 3 中的 transition 属性可以方便地实现属性之间的平滑过渡，是 JavaScript 之外的另一种实现方式。

2.19 图片翻转

使用 CSS 3，可以实现不借助其他的工具（JavaScript、Photoshop 等）轻松实现图片翻

转的功能，尽可能地提高了效率和图片的重复利用率。如图 2.39 所示，左右两张图片使用的是同一张图片素材，右边的图片是使用 CSS 制作的图片翻转效果。

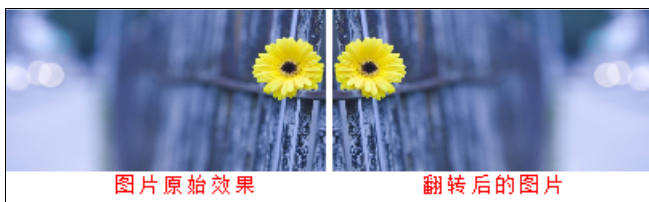


图 2.39 图片翻转

图片的翻转原理是 CSS 3 中变形属性中的 `scale`，当 `sacle` 取值为 `-1` 时，水平方向和垂直方向会同时得到翻转的效果，图中只是绕竖直方向翻转，所以只使用了 `scaleX`，代码如下：

```
01 .turn{
02     -moz-transform: scaleX(-1);
03     -o-transform: scaleX(-1);
04     -webkit-transform: scaleX(-1);
05     -ms-transform: scaleX(-1);
06     transform: scaleX(-1);
07     filter: FlipH;                      /*使用滤镜做兼容*/
08     -ms-filter: "FlipH";
09 }
```

由于代码单一使用 CSS 3 变形 `transform` 属性的使用方法，所以不再做解释，如有疑问可跳至第 5.1 节查看基本用法。

2.20 图像地图

图像地图是带有可点击区域的图像。通常情况下，每个区域是一个相关的超链接，单击某个区域，就会到达相关的链接。在 HTML 中，在 `img` 标签上可以使用 `usemap` 属性链接到一个已经包含 `area` 标签的 `map` 图片地图映射。如以下代码：

```
01     //img 使用地图 planetmap
02 <map name="planetmap" id="planetmap">                //planetmap 地图定义
03     <area shape="circle" coords="180,139,14" href="/example/html/venus.html"
04         target="_blank" alt="Venus" />                //圆形区域，圆心为（180,139），半径为 14
05     <area shape="circle" coords="129,161,10" href="/example/html/mercur.html"
06         target="_blank" alt="Mercury" />              //圆形区域，圆心为（129,161），半径为 10
07     <area shape="rect" coords="0,0,110,260" href="/example/html/sun.html"
08         target="_blank" alt="Sun" />                  //矩形区域，左上角（0,0），右上角（110,260）
09 </map>
```

虽然使用标签较为简便，但图片默认不会显示区域的提示，如果用户不知道图像上使用了地图映射，那么用户不会操作。为了增加提示，需要使用 JavaScript。其实使用元素的

定位在图片上增加浮动层，可以不需要任何 JavaScript 代码，并取得很好的视觉效果。如本实例的代码，这段代码的效果如图 2.40 所示。



图 2.40 图像地图

//HTML 代码

```
01 <div id="imagemap">
02 <span class='pc_s'></span><a href="#" class="pc_a">这是一台计算机</a>
03 <span class='kb_s'></span><a href="#" class="kb_a">这是键盘</a>
04 <span class='fl_s'></span><a href="#" class="fl_a">这盆景怎么样</a>
05 </div>
```

//CSS 代码

```
01 #imagemap{
02     width:500px;height:375px;
03     background:url("img/bg.jpg") no-repeat;
04     position:absolute;
05 }
06 .pc_s,.kb_s,.fl_s{
07     transition:all 1s;                                /*CSS 过渡效果，展示隐藏更平滑*/
08 }
09 -webkit-transition:all 1s;                            /*更多见第 5 章*/
10 -moz-transition:all 1s;
11 -o-transition:all 1s;
12 border:1px solid white;
13 display:block;position:absolute;
14 opacity:0;                                            /*鼠标不指向图片，区域框不显示*/
15 }
16 .pc_a,.kb_a,.fl_a{
17     padding:1%;
18     border:1px solid white;background:black;opacity:0; /*鼠标没有指向区域，说明不显示*/
19     text-decoration:none;color:#fff;
20     display:block;position:absolute;
21     transition:all 1s;                                /*CSS 过渡效果，展示隐藏更平滑*/
```

```

21     -webkit-transition:all 1s;
22     -moz-transition:all 1s;
23     -o-transition:all 1s;
24     border-radius:10px 10px;
25 }
26 .pc_s{width:20%;height:25%;top:16%;left:54%;}           /*各提示区域的大小设定*/
27 .kb_s{width:20%;height:11%;top:45%;left:54%;}
28 .fl_s{width:15%;height:28%;top:22%;left:11%;}
29 .pc_a{width:20%;height:8%;top:22%;left:77%;}           /*各链接部分的大小设定*/
30 .kb_a{width:20%;height:4%;top:48%;left:77%;}
31 .fl_a{width:20%;height:4%;top:13%;left:8%;}
32 #imagemap:hover .pc_s,#imagemap:hover .kb_s,#imagemap:hover .fl_s{
33     opacity:1;}                                           /*鼠标指向图片，区域框显示*/
34 .pc_s:hover+.pc_a{opacity:0.5;}                           /*鼠标指向区域框，说明框显示*/
35 .kb_s:hover+.kb_a{opacity:0.5;}
36 .fl_s:hover+.fl_a{opacity:0.5;}
37 .pc_a:hover,.kb_a:hover,.fl_a:hover{opacity:0.8;}       /*鼠标指向说明，区域背景加深*/

```

其中第 33~35 行中使用了 CSS 选择器的兄弟选择器“+”，提示区域和提示文字（链接部分）是同级别的元素，文字区域与提示区域是紧挨的元素，所以当鼠标悬浮在提示区域时，使用兄弟选择器可以直接匹配到对应的文字区域并对文字区域设定样式，通过透明度是文字区域展示给用户，从而无须 JavaScript 实现提示隐藏效果，这与第 5 章中的冒泡提示原理相同。

注意：兄弟选择器要求十分严格，只能匹配元素的直接下方元素，使用时往往采用独特的结构（必须使得元素紧挨），使得选择器匹配符能够正常使用。

第 3 章 按钮和链接



网站与访客的交互很大一部分是通过按钮和链接来实现的，好的按钮和链接样式能给访客赏心悦目的感受。在 CSS 2 时代里，一些好看的网页效果通常是由一些 Photoshop 处理过的图片组合出来的，而在 CSS 3 时代，新的特性使网页效果轻松实现，网页的体积更小、响应速度更快，却一样有着非凡的视觉体验。本章将重点讲述如何使用 CSS 对按钮和链接进行处理。

本章涉及的知识点有：

- 圆角按钮
- 简洁与滑动效果的导航菜单
- 下拉菜单、圆形导航菜单、网页右键菜单
- 标签云
- 选中文字分享
- 链接百叶窗效果
- iPhone 开关
- 按钮式单选框与复选框
- 自定义播放器
- 文字变链接、链接属性的顺序
- 根据文件格式设置链接图标

3.1 圆角按钮

按钮在网页上经常见，通常按钮可以有圆角、直角、平面、立体等。这些按钮，有的直接是 `<input type="button"/>` 的标签，有的则是某些 `a` 标签改造的。本节实现的效果是使用 CSS 3 分别基于 `input` 标签和 `a` 标签创造圆角立体按钮，效果如图 3.1 所示。由图像可以看出，`input` 标签和 `a` 标签通过 CSS 格式化之后长相并无明显区别，事实上，许多网页上的按钮是很难区分它们的本质的。

//HTML 部分

```
01 <input type="button" value="注册" class="round_button green"/>
```

```
02 <a href="#" class="round_button blue">登录</a>
```

在没有应用任何 CSS 之前，代码的执行效果如图 3.2 所示。

//CSS 部分

```

01 .round_button {
02   border: 1px solid;
03   display: block;
04   font: bold 12px/25px Arial, sans-serif;
05   font-size: 1em;
06   text-decoration: none;           /*文本无装饰*/
07   text-align: center;             /*文本水平居中*/
08   line-height: 50px;             /*文本垂直方向居中*/
09   width: 100px;
10   height: 50px;
11   margin: 10px;
12 }

```



图 3.1 效果展示



图 3.2 无 CSS

上述代码对两个元素做初步设置：第 1 行设置边框线为实线以便 CSS 编写过程中的定位（通常 border 属性会在 CSS 编写结束时删除），第 2 行把 a 标签设为块展示，第 4~5 行设置字体，第 6 行把 a 标签的下划线去掉，第 7~8 行使 a 标签的字体居中显示，最后第 9~11 行设定块的宽度、高度和边距。执行上述 CSS 之后的效果如图 3.3 所示。

```

01 .round_button {
02   -webkit-border-radius: 15px;
03   -moz-border-radius: 15px;       /*老版本的 Firefox 圆角*/
04   border-radius: 15px;
05 }

```

继续添加，使边框成为圆角，在 CSS 2 中做圆角的效果需要使用技巧，而在 CSS 3 中可以用 border-radius 轻松创建。border-radius 可以被 IE9+、Chrome 浏览器和 Opera 浏览器识别，-webkit-前缀使 Safari 浏览器识别，-moz-前缀使老版本的 Firefox 浏览器识别。执行上述代码后的效果如图 3.4 所示。

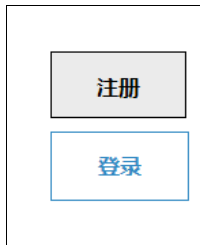


图 3.3 初步布局

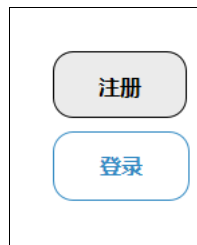


图 3.4 添加圆角

```

01 .green {
02   color: #3e5706;

```

```

03 background: #a5cd4e; /*纯色背景*/
04 background: -moz-linear-gradient(top, #a5cd4e 0%, #6b8f1a 100%); /*FF3.6+*/
05 background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#a5cd4e),
color-stop(100%,#6b8f1a)); /*Webkit*/
06 background:-webkit-linear-gradient(top,#a5cd4e 0%,#6b8f1a 100%);
/*Chrome10+,Safari5.1+*/
07 background: -o-linear-gradient(top, #a5cd4e 0%,#6b8f1a 100%); /*Opera*/
08 background: -ms-linear-gradient(top, #a5cd4e 0%,#6b8f1a 100%); /*IE10+*/
09 background: linear-gradient(top, #a5cd4e 0%,#6b8f1a 100%); /*W3C*/
10 }
11 .blue {
12 color: #19667d;
13 background: #70c9e3; /*纯色背景*/
14 background: -moz-linear-gradient(top, #70c9e3 0%, #39a0be 100%); /* FF3.6+ */
15 background:-webkit-gradient(linear,left top, left bottom,color-stop(0%,#70c9e3),
color-stop(100%,#39a0be)); /*Webkit*/
16 background: -webkit-linear-gradient(top, #70c9e3 0%,#39a0be 100%);
17 background: -o-linear-gradient(top, #70c9e3 0%,#39a0be 100%); /*Opera */
18 background: -ms-linear-gradient(top, #70c9e3 0%,#39a0be 100%); /* IE10+ */
19 background: linear-gradient(top, #70c9e3 0%,#39a0be 100%); /* W3C */
20 }

```

大体的框架出现后，设置元素的颜色，为了让颜色不单一并且增强立体感，对颜色的设置使用了 gradient 渐变。linear 渐变是线性渐变，对于支持 Chrome 浏览器和 Safari 浏览器来说，是 -webkit-gradient(渐变类型，起始位置，结束位置，color-stop(特定点位置 1，位置 1 颜色)，color-stop(特定点位置 2，位置 2 颜色)) 这种格式；对于其他浏览器来说，是前缀-linear-gradient(起始位置，特定点位置 1、位置 1 颜色，特定点位置 2、位置 2 颜色)) 上述代码的 0% 和 100% 也就是顶部和底部的位置（渐变更详细的要点在第 4 章的基础章节）。渐变做好后的效果如图 3.5 所示。

```

01 .round_button:hover {
02 -webkit-box-shadow: 1px 1px 1px rgba(0,0,0,.3), inset 0px 0px 2px rgba(0,0,0, .5);
03 -moz-box-shadow: 1px 1px 1px rgba(0,0,0,.3), inset 0px 0px 2px rgba(0,0,0, .5);
04 box-shadow: 1px 1px 1px rgba(0,0,0,.3), inset 0px 0px 2px rgba(0,0,0, .5);
05 }
06 .round_button:active {
07 -webkit-box-shadow: inset 0px 0px 3px rgba(0,0,0, .7); /*内阴影*/
08 -moz-box-shadow: inset 0px 0px 3px rgba(0,0,0, .7);
09 box-shadow: inset 0px 0px 3px rgba(0,0,0, .7);
10 }

```

大体的样式做好之后，通过给按钮添加鼠标悬浮 (:hover) 和鼠标单击 (:active) 的伪类和给边框添加内阴影 (box-shadow、inset) 的样式可以增强按钮的立体效果。第 2~4 行，当鼠标悬浮时，内边框阴影偏移 2 像素。第 7~9 行，当鼠标单击时偏移到 3 像素。box-shadow 的格式是：水平偏移、竖直偏移、阴影宽度、阴影颜色，inset 属性设置的是内边框阴影。与其他属性一样，box-shadow 也需要加前缀使浏览器识别。

```

01 .round_button {
02   -webkit-transition: all 0.15s ease;           /*CSS 过渡*/
03   -moz-transition: all 0.15s ease;
04   -o-transition: all 0.15s ease;
05   -ms-transition: all 0.15s ease;
06   transition: all 0.15s ease;
07 }

```

最后使用 `transition` 属性圆滑的过渡悬浮和单击事件改变的 CSS 属性。CSS 的 `transition` 允许 CSS 的属性值在一定的时间区间内平滑地过渡。这种效果可以在鼠标单击、获得焦点、被点击或对元素任何改变中触发，并圆滑地以动画效果改变 CSS 的属性值。`-webkit-transition: all 0.15s ease` 使得所有 CSS 属性在 0.15s 内越来越慢的转变并且不设置延迟。执行上述两段代码之后的效果如图 3.6 所示，这种效果在图片上不太明显，实际操作网页时可以看到明显效果。



图 3.5 设置渐变



图 3.6 最终效果

注意：CSS 3 的各种属性的使用方法穿插在各章节中，如 CSS 3 渐变出现在背景与颜色一章，CSS 动画出现在变化和动画一章，阴影出现在边框和图片一章等。在理解时可以转到属性详解的地点加以辅助理解。

第 2 章讲述边框属性时使用边框属性制作了圆角的按钮，与本方案相比使用更少的代码，也不失为一种好的方案。

3.2 简单导航栏

导航栏多见于网页的头部区域，是一个链接集合的区块，使访客快捷地跳转到不同的模块。通常来说，在一个网页里，三级导航已经足够使用，三级以上的导航栏不常见。经常使用的是一级导航和二级导航。由于导航栏的实现方法多种多样，所以导航栏按实现方法和效果分类会分布在不同的章节中。本节是一级导航栏的实现，效果如图 3.7 所示。

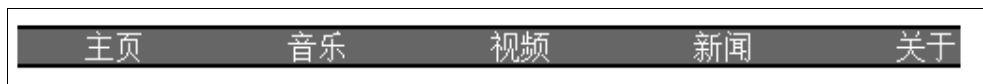


图 3.7 一级导航栏

纯 CSS 代码方法的原理是利用一个无序列表的 CSS 定位和 CSS 伪类操作。

```

01 <div class="nav_1">
02 <ul>
03 <li><a href="#">主页</a></li>
04 <li><a href="#">音乐</a></li>
05 <li><a href="#">视频</a></li>
06 <li><a href="#">新闻</a></li>
07 <li><a href="#">关于</a></li>
08 </ul>
09 </div>

```

上面无任何 CSS 的 HTML 代码执行效果如图 3.8 所示。

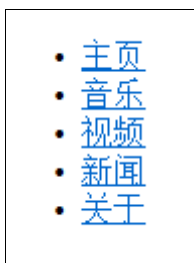


图 3.8 一级导航初始效果

```

//CSS 代码
01 .nav_1{
02     margin:50px auto;
03     background-color:#666;
04     border-top:2px solid #000;           /*只设置上边框和下边框*/
05     border-bottom:2px solid #000;
06 }
07 .nav_1>ul{
08     margin:0 auto;
09     list-style:none;                     /*去除列表符号*/
10     position:relative;
11     width:960px;
12 }
13 .nav_1>ul>li>a{
14     display:block;
15     width:100px;
16     color:#FFFFFF;
17     text-align:center;
18     text-decoration:none;
19 }

```

首先是对导航栏整体的大小、背景等基本属性的设置。第 2 行和第 8 行是整体居中的实现。第 4~5 行设置导航栏上下的边框为 2 像素。第 9 行把无序列表的项目符号去掉。第 14 行使链接作为块级元素显示，第 17 行使链接文本居中显示，第 18 行把链接的下画线去掉。代码运行后的结果如图 3.9 所示。可以看到做到这一步时一个竖直方向的导航栏已经

做好了。本节中我们需要的是一个水平导航栏，所以还需要把链接放在同一行显示。在 CSS 中加以下代码：

```
01 .nav_1>ul>li{
02     display:inline-block;
03 }
```

再次运行代码，即可得到最终的效果。



图 3.9 初步设置

3.3 二级导航栏

二级导航的效果如图 3.10 所示，当鼠标指向某一区块，就会显示二级菜单。

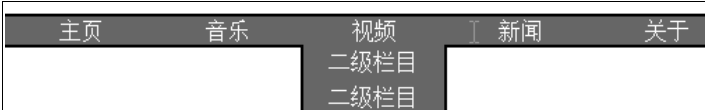


图 3.10 二级导航

与 3.2 节相似，下面是二级导航的 HTML 代码结构，在做二级导航时，需要在一级导航列表中定义子列表。

```
//二级导航 HTML 结构
01 <div class="nav_2">
02   <ul>
03     <li><a href="#">主页</a>
04     <ul>
05       <li><a href="#">二级栏目</a></li>
06     </ul>
07   </li>
08 </ul>
09 </div>
```

这种结构的代码运行效果如图 3.11 所示。同一级导航的原理，使用以下 CSS 代码对其做初步设置，处理后的效果如图 3.12 所示。

```
01 .nav_2{
02     margin:50px auto;
03     height:20px;
04     background-color:#666;
05     border-top:2px solid #000;           /*只设置上边框和下边框*/
06     border-bottom:2px solid #000;
07 }
08 .nav_2>ul{
09     margin:0 auto;
10     list-style:none;                   /*去除列表符号*/
```

```
11 position:relative; /*保留在文档流中，供二级导航绝对定位*/
12 width:960px;
13 }
14 .nav_2>ul>li{
15     float:left; /*列表项目左浮动排列*/
16     line-height:20px;
17 }
18 .nav_2>ul>li>a{
19     display:block;
20     width:105px;
21     color:#FFFFFF;
22     text-align:center;
23     text-decoration:none;
24 }
25 .nav_2>ul>li>ul{ /*二级 ul*/
26     list-style:none;
27 }
28 .nav_2>ul>li>ul>li{
29     display:block;
30     background-color:#666;
31     line-height:25px;
32     border-left:2px solid #000;
33     border-right:2px solid #000;
34 }
35 .nav_2>ul>li>ul>li>a{
36     display:block;
37     width:100px;
38     color:#FFFFFF;
39     text-align:center;
40     text-decoration:none;
41 }
```

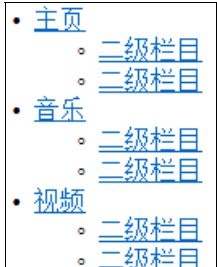


图 3.11 二级导航无 CSS 效果

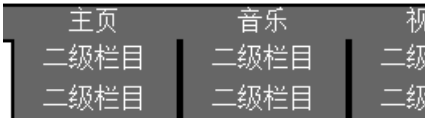


图 3.12 二级导航初步设置效果

下面还需要做的是让二级栏目的内容默认不显示，当鼠标移动到一级栏目上的时候显示出相对应的二级栏目内容，需要添加以下 CSS 代码：

```
01 .nav_2>ul>li>ul{
02     display:none;
```

```

03 }
04 .nav_2>ul>li:hover >ul{           /*鼠标指向时显示二级列表*/
05     display:block;
06 }

```

再次运行代码即可得到所需效果。

3.4 三级导航栏

有了上面的一级导航和二级导航之后，做三级导航就容易了，原理也是相同的。HTML 结构是在二级导航的列表下再增加一次无序列表。HTML 结构如下，效果如图 3.13 所示。

```

01 <div class="nav_3">
02 <ul>
03     <li><a href="#">主页</a>
04     <ul>
05         <li><a href="#">二级栏目</a>
06         <ul>
07             <li><a href="#">三级栏目</a></li>
08         </ul>
09     </li>
10 </ul>
11 </div>
12 </div>

```

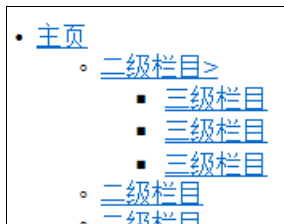


图 3.13 三级导航无 CSS 效果

使用以下的 CSS 代码对三级导航做初步设置，为了节约篇幅，其中与一级导航和二级导航同理的代码已经省略。

```

01 .nav_3>ul>li>ul>li>ul{
02     list-style:none;
03     border-top:2px solid #000;
04     border-right:2px solid #000;
05     border-bottom:2px solid #000;
06     background-color:#666;
07     position:absolute;           /*相对父级元素依然是绝对定位的*/
08     left:103px;

```



```

09  top:0;
10 }
11 .nav_3>ul>li>ul>li>a{                /*三级列表链接设置*/
12  display:block;
13  width:100px;
14  color:#FFFFFF;
15  text-align:center;
16  text-decoration:none;
17 }

```

请注意这里的第 7 行，这里是三级导航与其他不同的地方，在对三级栏目做 position 定位时，由于 absolute 使该内容脱离文件流而相对于父容器定位，如果不设置二级无序列表的 li 项定位为 relative 定位，那么三级栏目就不会相对于二级栏目来定位而是相对于一级导航来进行定位。运行上面的代码，可以看到初步设置后的效果如图 3.14 所示。这时还达不到所需的效果，还需要使二级以上的导航默认不显示，鼠标移动到上一级的列表项时触发显示，所以需要以下代码。

```

01 .nav_3>ul>li>ul{
02  display:none;
03 }
04 .nav_3>ul>li>ul>li>ul{
05  display:none;
06 }
07 .nav_3>ul>li:hover >ul{                /*显示二级列表*/
08  display:block;
09 }
10 .nav_3>ul>li:hover >ul>li:hover >ul{    /*显示三级列表*/
11  display:block;
12 }

```

再次运行代码，即可得所需效果，如图 3.15 所示。

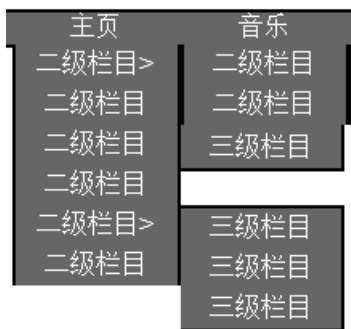


图 3.14 初步设置后的效果

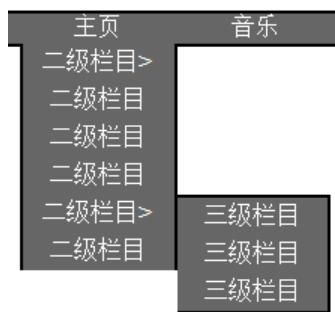


图 3.15 最终效果

注意：本节只是用 CSS 代码来实现导航栏，这种方法的关键在于对元素的定位以及 CSS 伪类的使用，实例偏重原理实现，通常与 JavaScript 或者与 jQuery 配合使用，制作更加强大的菜单导航，因为 JavaScript 可以更方便快捷地控制网页效果。

3.5 滑动菜单

滑动菜单能够使简单的导航菜单变得生动，图 3.16 为滑动菜单效果图，当鼠标指向导航栏的某一链接时，链接下方的小块就会移动到鼠标所指的链接下方，并且颜色会随机改变。在本节实例中，样式能够随链接的数量自适应达到最好的效果。



图 3.16 滑动菜单

本效果主要运用元素定位和元素运动实现，最外层为一个 div 容器，内层是一个包含 li 的 ul，li 中为 a 标签。CSS 部分的代码如下：

```
01  *{margin:0;padding:0;}
02  #ani_links{
03      width:800px;height:30px;
04      position:absolute;top:100px;left:200px;
05      border-bottom:1px solid;           /*下方横线*/
06  }
07  .ani_links {
08      width:100%;height:100%;
09      list-style:none;                   /*列表基本设置*/
10      position:absolute;
11      color:#FFFFFF;
12  }
13  .ani_links>li{text-align:center;float:left;background:#808080;} /*同排展示*/
14  .ani_links>li>a{text-decoration:none;font-size:16px;display:block;color:#FFFFFF;}
```

在 CSS 中指定外层 div 的大小及位置（第 3~4 行），通过整体的大小使用 JavaScript 动态确定每个 li 最合适的大小。除去最后一个 li 使用绝对定位（将在 JavaScript 中改变定位方式），其余的 li 元素按照左浮动的方式定位（第 13 行）。JavaScript 部分通过改变最后一个 li 的 left 值而改变它的位置。JavaScript 的实现原理如下：

```
01  <script type="text/javascript" src="move.js"></script>           //加载运动框架
02  <script type="text/javascript">
03      function getbyclass(parent,classname){                       //通过类名获取元素
04          var result=new Array();
05          var allclass=parent.getElementsByTagName("*");
06          for (var i=0; i<allclass.length;i++ )
07          {
08              if(classname==allclass[i].className)
09                  result.push(allclass[i]);
10          }
```

```

11 return result;
12 }
13 function color(){ //随机生成颜色
14   var r=parseInt(Math.random()*255);
15   var g=parseInt(Math.random()*255);
16   var b=parseInt(Math.random()*255);
17   var newcolor="rgb("+r+","+g+","+b+")";
18   return newcolor;
19 }
20 window.onload=function (){ //页面加载完成
21   var ani_links=document.getElementById('ani_links'); //获取最外层 div
22   var ani_links_ul=getbyclass(ani_links,'ani_links')[0]; //获取 ul
23   var ani_links_ul_width=ani_links_ul.offsetWidth; //获取 ul 宽度
24   var ani_links_ul_height=ani_links_ul.offsetHeight; //获取 ul 高度
25   var ani_links_lis=ani_links_ul.getElementsByTagName('li'); //获取 li
26   var ani_links_li_border=1;var bottom=3;var bottom_left=0;
27   //li 的边框、下方可滑动控件的高度、可滑动控件的 left 值变量
28   var n=ani_links_lis.length; //获取 li 的数量
29   var ani_links_li_width=Math.floor(ani_links_ul_width/(n-1)); //计算每个 li 所占空间的平均宽度
30   for (var i=0;i<n-1;i++ )
31   {
32     ani_links_lis[i].style.width=ani_links_li_width-ani_links_li_border*2+"px";//li 元素的宽
33     ani_links_lis[i].style.height=ani_links_ul_height-ani_links_li_border*2-bottom+"px"; //li 高
34     ani_links_lis[i].style.lineHeight=ani_links_ul_height-ani_links_li_border*2-bottom+"px";
35     ani_links_lis[i].style.border=ani_links_li_border+"px solid"; //设置 li 的边框
36     ani_links_lis[i].onmouseover=function (){ //鼠标指向链接
37       bottom_left=this.offsetLeft; //获取当前元素相对于父元素的左偏移量
38       var c=color(); //随机生成一种颜色
39       move(ani_links_lis[n-1],{left:bottom_left},"flex");
40     } //改变可滑动控件位置，move 函数为 move.js 中的函数，本行可使元素按照弹性运动方式运动
41     ani_links_lis[n-1].style.backgroundColor=c; //设置颜色
42   }
43 }
44 ani_links_lis[n-1].style.position="absolute"; //初始化滑动控件
45 ani_links_lis[n-1].style.bottom=0;
46 ani_links_lis[n-1].style.left=0;
47 ani_links_lis[n-1].style.width=ani_links_li_width+"px";
48 ani_links_lis[n-1].style.height=bottom+"px";
49 ani_links_lis[n-1].style.background="red";
50 }
51 </script>

```

第 39 行中的 move 函数为运动框架中的内容，作用与 jQuery 中的 animate 方法相似，函数的使用可以查看实例代码中的 move.js，这里不做解释。

3.6 网页右键菜单

网页右键菜单在网络上应用并不普遍，这种效果可以被应用在 Web APP 上，比如腾讯的 Web QQ，当在网页上单击右键时，可以弹出漂亮的菜单。由于多数网站并不偏向于 Web APP，有的网站使用 Flash 实现，所以并不是经常见到。本节的右键菜单在样式上逊色，重在 CSS 实现与 JavaScript 实现的过程，效果如图 3.17 所示。

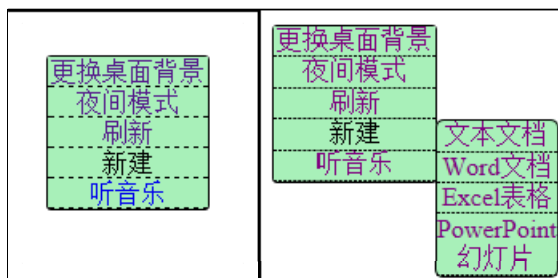


图 3.17 网页右键菜单

网页菜单的原型为连接导航栏，从图中的效果来看，这就是一种竖直方向的导航菜单而已。当导航菜单的展示与单击右键结合起来，就会变成右键菜单。首先是 HTML 部分的结构，在 a 标签里，既可以指向某一页面，又可以指向某一 JavaScript 功能，对应在页面上，一种是链接，一种是按钮。

```

01 <div id="right_button_menu">
02 <ul>
03 <li><a href="">更换桌面背景</a></li>
04 <li><a href="">夜间模式</a></li>
05 <li><a href="">刷新</a></li>
06 <li>
07 <ul>
08 <li><a href="">文本文档</a></li>
09 <li><a href="">Word 文档</a></li>
10 <li><a href="">Excel 表格</a></li>
11 <li><a href="">PowerPoint 幻灯片</a></li>
12 </ul>新建
13 </li>
14 <li><a href="javascript:show('player');">听音乐</a></li>
15 </ul>
16 </div>

```

第 7~12 行为二级菜单，考虑到定位，该二级菜单的上级名称写在该二级 ul 的后边，这样 CSS 代码将会变得简单。外层的 ul 为一级菜单，每个 li 为一个一级栏目，对于菜单来说，二级菜单已经足够。使用 CSS 布局如下：

```

01 *{margin:0;padding:0;}
02 #right_button_menu{
03 width:100px;height:auto;border:1px solid;border-radius:3px;

```

/*菜单*/

```

04 position:absolute;display:none; z-index:100;           /*默认不显示*/
05 }
06 #right_button_menu>ul{                                  /*列表*/
07   position:relative;background:hsl(134,70%,80%);
08 }
09 #right_button_menu>ul>li{                                /*同导航栏的实现*/
10   border-bottom:1px dashed;position:relative;
11   text-align:center;list-style:none;cursor:pointer;
12 }
13 #right_button_menu>ul>li>a{
14   text-decoration:none;
15 }
16 #right_button_menu>ul>li>ul{
17   position:absolute;left:100px;
18   border:1px solid;border-radius:5px ;
19   display:none;background:hsl(134,70%,80%);
20 }
21 #right_button_menu>ul>li>ul>li{
22   border-bottom:1px dashed;
23   text-align:center;list-style:none;
24 }
25 #right_button_menu>ul>li>ul>li>a{
26   text-decoration:none;
27 }
28 #right_button_menu>ul>li: hover>ul{                     /*展示菜单的内容*/
29   display:block;
30 }

```

在本实例中，使用 `hover` 伪类制作二级菜单的弹出效果，JavaScript 只管右键菜单的整体位置。默认状态下菜单是不可见的，所以在第 4 行隐藏此元素，当右键点击时，菜单出现在鼠标的位置处，所以它使用绝对定位并且弹出时应该处于网页的最上层。对于每个一级项目，设置为相对定位（第 10 行）以便二级 `ul` 能以他们为父元素做绝对定位（第 17 行），定位的 `left` 值为一级项目的宽度。二级菜单也是不展示的，只有当鼠标停留在一级项目上方才展示出来（第 19 行和第 29 行）。这样一个位置固定的菜单就完成了，其余工作交给 JavaScript 来做。

JavaScript 的主要工作有：禁用网页默认的鼠标右键菜单、当鼠标右键被按下时获取到当前鼠标的位置并把菜单展示在此处、再次单击鼠标左键时隐藏菜单或者执行菜单的功能、再次单击右键时重新调整菜单到当前的位置。

```

01 window.onload =function (){
02   var menu=document.getElementById('right_button_menu'); //获取到菜单
03   document.oncontextmenu=function(e){return false;} //禁止火狐、IE、safari 右键菜单弹出
04   function click(e) {                                     //点击处理函数
05     var e=e||event;                                     //为了兼容

```

```

06     if (e.which==2||e.button==4) {                                //中键
07         hide_menu(menu);                                          //隐藏菜单
08         return false;
09     }
10     else if(e.which==3||e.button==2){                             //右键
11         x=e.clientX;y=e.clientY;                                  //获取鼠标的位置
12         var l=document.body.scrollLeft>0?document.body.scrollLeft:
13             document.documentElement.scrollLeft;                //获取鼠标位置的左偏移
14         var t=document.body.scrollTop>0?document.body.scrollTop:
15             document.documentElement.scrollTop;                  //获取鼠标位置的上偏移
16         document.title="X 坐标:"+(x+l)+" ,Y 坐标:"+(y+t);        //在标题上显示 left 和 top
17         show_menu(menu,x +l,y+t);                                //展示菜单
18     }
19     else if(e.which==1||e.button==1){                             //左键
20         setTimeout(function (){hide_menu(menu);},100);
21 //隐藏菜单，如果点击菜单内容，设置一个延迟让菜单的功能实现，否则在 IE 下来不及实现
22     }
23 }
24 document.onmousedown=click;                                       //文档鼠标按下执行处理
25 function show_menu(obj,left,top){                                  //展示函数
26     obj.style.left=left+'px';
27     obj.style.top=top+'px';
28     obj.style.display='block';
29 }
30 function hide_menu(obj){                                          //隐藏函数
31     obj.style.display='none';
32 }
33 }

```

注意：原生 JavaScript 实现时，要想获得绚丽的动画效果，需使用元素运动的相关操作，使用 jQuery 可以方便快捷的制作淡入淡出、缓慢下拉、缓慢收起等效果。实现的工作同上述的介绍，在实例代码中附带 jQuery 实现的动态右键菜单，文章中不再赘述。

3.7 下拉菜单

之前的章节分别从纯 CSS 方面和与 JavaScript 结合的方面制作了不同实现的导航栏和菜单。CSS 3 能带给网页效果很多改变，本节的内容是使用 CSS 3 对一个 div 和一个 ul 美化而成的漂亮下拉菜单，它拥有 CSS 3 的优点——对 JavaScript 的依赖性减弱。如图 3.18 所示，这是一个简洁的下拉菜单，可以以本段代码为基础扩展出更多的展示效果。



图 3.18 下拉菜单

在 HTML 部分实例并没有选择 select，下面是结构：

```
01 <div class="fleft" id="menu">
02   <div class="cd-dropdown"><span>将此内容分享到</span>
03   <ul>
04     <li><span class="icon-google-plus">Google Plus</span></li>
05     <li><span class="icon-facebook">Facebook</span></li>
06     <li><span class="icon-twitter">Twitter</span></li>
07     <li><span class="icon-github">GitHub</span></li>
08   </ul></div>
09 </div>
```

对整体和第一张选项卡的设置，包括第一张选项卡的尺寸、阴影和字体设置。

```
01 *{margin:0;padding:0;}
02 body{background:url("img/bg.jpg");}
03 .fleft{width:200px;margin:100px auto;position:relative;}
04 .cd-dropdown{
05   width:300px;height:30px;background:#fff;
06   box-shadow:1px 1px 1px 2px #ccc;
07   margin:10px;padding:10px;line-height:30px;text-align:center;
08   font-size:20px;font-weight: bold;color:hsla(223,50%,45%,0.7);
09   cursor:pointer;
10 }
```

使用 after 伪对象和 border 在第一张选项卡上生成一个三角（第 2 章有述）。

```
01 .cd-dropdown:after{
02   content:"";
03   top:20px;right:-100px;position:absolute;
04   border-top:5px solid black;
05   border-left:5px solid transparent;
06   border-right:5px solid transparent;
07   border-bottom:5px solid transparent;
08 }
```

对弹出选项卡的设置，其中第 8~12 行指定了旋转变换的基点（绕该点旋转），第 11~17

行为指定过渡效果（可参照第 5 章）。

```

01 .cd-dropdown ul{position:absolute;left:0;margin-top:10px;}
02 .cd-dropdown li{
03   width:300px;height:30px;background:#fff;
04   position:absolute;top:-30px;opacity:0;
05   box-shadow:1px 1px 1px 2px #ccc;
06   margin:10px;padding:10px;line-height:30px;text-align:center;
07   list-style:none;cursor:pointer;
08   transform-origin:10% 50%;           /*变换基点的设置，第 5 章中有详解*/
09   -ms-transform-origin:10% 50%;
10   -webkit-transform-origin:10% 50%;
11   -o-transform-origin:10% 50%;
12   -moz-transform-origin:10% 50%;
13   -webkit-transition: all 0.2s linear 0s;   /*CSS 过渡，平滑切换，第 5 章中有详解*/
14   -moz-transition: all 0.2s linear 0s;
15   -ms-transition: all 0.2s linear 0s;
16   -o-transition: all 0.2s linear 0s;
17   transition: all 0.2s linear 0s;
18 }
19 .cd-dropdown li:hover{background:hsla(153,50%,45%,0.7);}/*改变背景颜色*/

```

通过 before 伪对象在选项卡上生成图标，使用文字作为图标有很大的优越性。

```

01 @font-face{                               /*使用文字取代图标*/
02   font-family:icon;
03   src:url("css/fonts/icon.eot"),url("css/fonts/icon.svg"),
04        url("css/fonts/icon.ttf"),url("css/fonts/icon.woff");
05 }                                           /*图像代码与字体文件有关，下方 content 属性值*/
06 .icon-google-plus:before{font-family:icon;content: "\21";font-size:40px;}
07 .icon-facebook:before {font-family:icon;content: "\22";font-size:40px;}
08 .icon-twitter:before {font-family:icon;content: "\23";font-size:40px;}
09 .icon-github:before {font-family:icon;content: "\24";font-size:40px;}

```

布局到此就做好了，应用 CSS 3 过渡效果可以使 JavaScript 部分的代码只关注最终的值，而渐变的动画会由 CSS 3 自动完成。JavaScript 需要做的工作主要有：获取第一张选项卡并为其添加鼠标点击事件，使用模运算判断点击后的折叠或展开行为，随机生成选项卡的旋转角度等，以下为 JavaScript 代码：

```

01 window.onload =function (){
02   var all=document.getElementById('menu');           //整体
03   var button=all.getElementsByTagName('div')[0].getElementsByTagName('span')[0]; //按钮
04   var oul=all.getElementsByTagName('ul')[0];        //列表
05   var lis=oul.getElementsByTagName('li');            //选项卡
06   var click=0;                                       //点击计数器
07   button.onclick=function (){
08     click++;                                         //计数器+1
09     if (click%2)                                    //判断点击

```



```

10 {
11   for (var i=0;i<lis.length ;i++ )
12   {
13     lis[i].style.top=i*(lis[i].offsetHeight+10)+"px";           //定位高度
14     lis[i].style.opacity=1;                                       //透明度
15     lis[i].style.transform="rotate("+Math.floor(Math.random()*5)+"deg)";
16                                     //随机生成 5 度以内的数字并应用在旋转
17   }
18 }else{
19   for (var i=0;i<lis.length ;i++ )
20   {
21     lis[i].style.top=0+"px";                                       //恢复初始值
22     lis[i].style.opacity=0;
23     lis[i].style.transform="rotate("+0+"deg)";
24   }
25 }
26 };
27 }

```

注意：CSS 本身就是一种堆叠性的语言，一个好的网站往往都是在创意上取胜的。本实例仅起抛砖引玉的作用，由本实例扩展更多的展示效果，丰富的创意也能在设计中略胜一筹。

3.8 CSS 3 圆形导航菜单

本节的实例为使用 CSS 3 2D 转换 transform 中 translate 属性制作的圆形导航菜单，该效果应用在页面的左下角，当点击左下角的加号图标时，加号右上方的 5 个图标将会依次弹出，并做完弹性运动后停留在图中位置。当再次点击加号图标时，5 个图标将同时回到加号图标的下方，动画效果十分华丽。本实例重在说明实现原理，在合适大小的范围内，最多能够容纳的图标总数一定（即图标特别多的情况下不能很好地适应，直接使用时需要控制图标的数量，也可根据原理和实际情况量身定做），实例效果如图 3.19 所示。



图 3.19 CSS 3 圆形导航菜单

在 CSS 布局上，使用到 CSS 3transform 属性中的 translate 平移变换。translate 变换与 position 定位的本质不同，CSS 3 中的 2D、3D 转换的原理为矩阵变换，更加适应动画的需求，而 position 定位仅仅为了解决布局的需求。两者相比，CSS 3 中的变换更加优秀。

translate 的语法为（更多详情请转到第 5.1 节）：

```
transform:translate(x,y);
```

参数 x 为横轴 x 轴的偏移量，参数 y 为纵轴 y 轴的偏移量，偏移量单位 px 需一同写入参数中。在 IE10+ 以外的浏览器下，需要加前缀才有效果。如 -webkit-、-o-、-moz-，IE9 需加 -ms- 前缀。对应的脚本特性为 transform，如 object.style.transform=translate('10px 10px '), 在需要前缀的浏览器下，对应的脚本特性为 webkitTransform、OTransform、MozTransform 和 msTransform。也就是说，通过 JavaScript 可以改变 CSS 属性，由于 translate 使用的参数与数值有关，这也为元素的运动提供了可能。本实例中的 translate 运动的实现函数在运动框架中，是本例在运动方面的基础函数，但由于取值与实现烦琐（且使用 translate 在实现原理上的矩阵变换 matrix 的本质），在这里不对运动框架中的函数做解释，但可以通过查看实例代码理解。在 JavaScript 中遇到 move 函数时，仅需知道 move 函数为运动函数即可。

实例的 HTML 代码和 CSS 代码如下：

```
//HTML 代码
01 <div id="round_nav">
02 <ul class="round_nav">
03     6 个 li

//CSS 代码
01 *{margin:0;padding:0;}
02 #round_nav{
03     width:200px;height:200px;
04     position:absolute;left:50px;bottom:30px;
05 }
06 .round_nav{
07     list-style:none;
08     position:relative;
09 }
10 .round_nav>li{
11     width:50px;height:50px;
12     position:absolute;
13     -webkit-transform:translate(50px,150px);           /*CSS 平移属性*/
14     -moz-transform:translate(50px,150px);
15     -o-transform:translate(50px,150px);
16     -ms-transform:translate(50px,150px);
17     transform:translate(50px,150px);
18 }
```

在 CSS 中第 13~17 行将所有的 li 位置移动到外层 div 的左下角位置，并为 JavaScript 中属性值得改变做准备。其余的工作则通过 JavaScript 动态完成：

```
01 <script type="text/javascript" src="zQuery.js"></script>           //引入运动框架
```

```

02 <script type="text/javascript">
03 window.onload=function(){
04   var round_links=document.getElementById('round_nav');           //获取最外层 div
05   var round_links_ul=getbyclass(round_links,"round_nav")[0];       //获取 ul
06   var round_links_lis=round_links_ul.getElementsByTagName('li');   //获取 li
07   var n=round_links_lis.length;                                     //获取 li 的个数
08   var r1=round_links.offsetWidth;var r2=round_links_lis[0].offsetWidth; //外层的宽及 li 的宽
09   round_links_ul.style.left=-r2+"px";                             //修正 ul 的相对定位
10   var adeg=90/n;                                                  //计算右上两元素之间的度数
11   for (var i=0;i<n ;i++ )
12   {
13     round_links_lis[i].style.background="url(img/"+(i+1)+".png) no-repeat"; //设置背景图片
14   }
15   move(round_links_lis[0],{translate:[r2,r1-r2]},"flex");         //将第一个元素移动
16   round_links_lis[0].style.zIndex=5;                             //设置 z-index
17   var click=1;                                                    //点击次数存放变量
18   round_links_ul.onclick=function (){
19     if(click%2){                                                  //点击奇数次，展开菜单
20       round_links_click();click++;
21     }else{                                                        //点击偶数次，收回菜单
22       round_links_dbclick();click++;
23     }
24   }
25   function round_links_click(){                                   //展开菜单函数
26     var i=1;
27     var t=setInterval(function (){                                //定时器依次弹出每个元素
28       var deg=adeq*i/180*Math.PI;                                //计算元素与竖直方向的角度
29       var h=r1-r1*Math.sin(deg);var w=r1*Math.cos(deg);         //通过角度计算参数值
30       //将当前元素通过“flex”弹性运动的方式，将元素运动至 translate 值为[w,h]处
31       move(round_links_lis[i],{translate:[w,h]},"flex");
32       i++;
33       if(i==n){                                                  //所有元素弹出完毕
34         i=1;
35         clearInterval(t);
36       }
37     },150);
38   }
39   function round_links_dbclick(){                                  //收回菜单函数
40     for (var i=1;i<n;i++)
41     {
42       move(round_links_lis[i],{translate:[r2,r1-r2]},"flex");     //将元素运动回左下角
43     }
44   }
45 }
46 </script>

```

在理解 JavaScript 第 31 行的参数计算，可以参照图 3.20。通过图中的长度值及三角函数值，即可计算出 `translate` 所需的参数值。

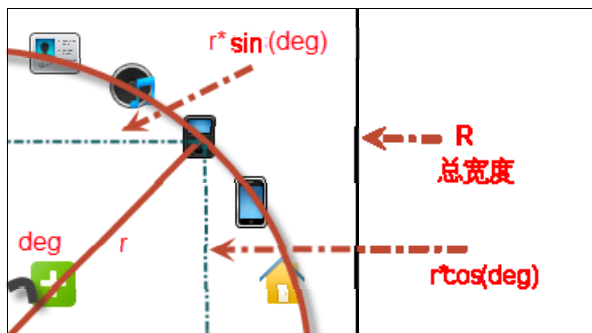


图 3.20 参数计算

注意：在一步步地设计时，最好先让各元素的边框可见，便于查看定位效果。所有定位完成之后再去掉边框。

本实例的效果也可以单纯地使用定位实现，实现的原理大致相同，但需计算各元素的定位属性值。虽然在 `li` 数目固定时较为快速，但当 `li` 数目不固定时，在 JavaScript 动态计算方面实现存在不足。同时因本书偏重新版本的 CSS 应用，所以单纯定位的代码不再实现。

3.9 标签云

标签云多用在个人博客上，较为常见的是 WordPress 上的应用。平面标签云的实现虽然较为简单，但作为链接中常用的一种形式，所以在本节做简单的介绍。标签云的效果如图 3.21 所示，它主要使用随机的文字大小和字体颜色对不同的链接设置不同的样式。

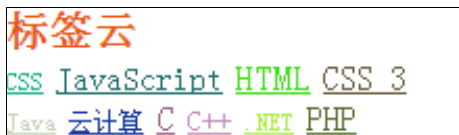


图 3.21 标签云

CSS 和 JavaScript 所有代码如下：

```
//CSS 代码
01 #tagcloud{width:200px;border:1px solid;margin:0;padding:0;}
02 #tagcloud p{font-size:18px;margin:0;padding:0;font-weight:bold;color:#ed5412}
//JavaScript 代码
01 window.onload =function (){
02   var tagcloud=document.getElementById('tagcloud');
03   var oa=tagcloud.getElementsByTagName('a');
04   for (var i=0;i<oa.length ;i++ )
05   {
```

```

06 oa[i].style.fontSize=(Math.random()+0.8)+'em';
07 oa[i].style.color="rgb("+parseInt(Math.random()*255)+","
08           +parseInt(Math.random()*255)+
09           ","+parseInt(Math.random()*255)+")";}}

```

3.10 TAB 标签页

与图片轮播相似，因可以实现最大程度扩展页面的可用空间，TAB 标签页应用比较广泛。TAB 标签页的实现一般是使用 z-index 来实现，也有使用左右滑动的效果。标签页的样式虽多，实现过程却是简单的。在 CSS 3 之前，制作标签页大多是通过 JavaScript 控制 z-index 实现，除了这种原理，还可以使用 CSS 3 中的 target 伪类来实现。图 3.22 为简洁的 TAB 标签页效果图。

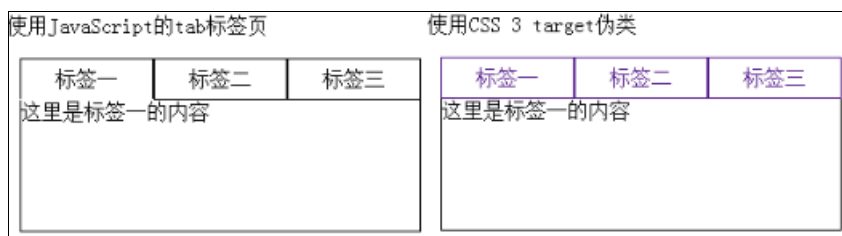


图 3.22 TAB 标签页

在本节的实例中，使用 JavaScript 实现与使用伪类实现在 CSS 布局上是一致的。整个标签页是一个整体，它包含 3 个左浮动 a 标签和 3 个绝对定位重叠在一起的 div。当上方的标签被点击时，就把下方对应的 div 调整到最上层。CSS 部分的代码如下，其中两中不同实现的 div 有着相同的布局。

```

01 #tab,#tab2{
02     width:310px;height:130px;
03     margin:10px;position:relative;    /*absolute 也可，下级 div 根据父元素绝对定位*/
04 }
05 a{
06     width:100px ;height:30px ;
07     border:1px solid;
08     display:block;cursor:pointer;float:left;    /*标签左浮动*/
09     text-decoration:none;text-align:center;line-height:30px;
10 }
11 #tab>div,#tab2>div{
12     border:1px solid;border-top:none;
13     width:304px ;height:100px;
14     position:absolute;top:32px;    /*脱离文档流后不能被标签盖住*/
15     background:#FFFFFF;    /*背景要指定，否则 div 中的内容会叠加在一起*/
16 }

```

3.10.1 使用 JavaScript

在上方标签被点击时，要通过 JavaScript 判断是第几个标签被点击，然后把下方的第几个 div 通过 zIndex 调整到上方。这一数字，必然与 index 值相等，JavaScript 并没有获取元素索引值的函数，需要执行设计。JavaScript 部分的代码为：

```

01 function index(current, obj) //获取元素索引值，参数为一般为 this,元素所在的集合元素
02 {
03     for (var i = 0; i < obj.length; i++)
04     {
05         if (obj[i] == current) return i;
06     }
07 }
08 window.onload=function(){
09     var tab=document.getElementById('tab');
10     var tab_a=tab.getElementsByTagName('a');
11     var tab_content=tab.getElementsByTagName('div');
12     for (var i=0;i<tab_a.length ;i++ )
13     {
14         tab_a[i].onclick =function () {show_tab_content(index(this,tab_a))}
15     }
16     function show_tab_content(i){ //展示 div
17         for (var j=0;j< tab_content.length ;j++ )
18         {
19             tab_a[j].style.borderBottom="1px solid"; //所有标签的下边框显示
20             tab_content[j].style.zIndex=0; //所有 div 的 z-index 值为 0
21         }
22         tab_a[i].style.borderBottom='0'; //当前的标签下边框为 0
23         tab_content[i].style.zIndex=2; //当前的 div 的 z-index 值为 2
24     }
25 }

```

3.10.2 使用 CSS target 伪类

与 JavaScript 相比，target 伪类实现更加简单。target 为目标伪类，它的语法是 E:target{描述}，它的作用是匹配 url 中指向的 E 元素，并把 CSS 代码应用到 E 元素上。如 #test:target{background:red}，当在当前网址的后面输入 #test 时，浏览器就会查找到该页面上 id 为 test 的元素，并把该元素的背景设置为红色。对于 tab 页上的多个重叠的 div，将其用 target 伪类匹配并应用到 z-index:2，即可把该 div 调整到最上层。因为用户不会主动地在地址栏输入 #id，所以该工作交给 a 标签，如 ，所以在实例中，HTML 部分为：

```

01 <div id="tab2">
02 <a href="#div1">标签一</a><a href="#div2">标签二</a><a href="#div3">标签三</a>

```



```
15 t+=ev.clientY; //计算分享按钮的 top 值
16 if(select().length>10){
17     setTimeout(function (){ //解决 IE 下文本选中不正确的问题
18         share_button.style.display='block';
19         share_button.style.left=1+'px';
20         share_button.style.top=t+'px';
21     },100);
22 }else{
23     share_button.style.display='none'; //小于 10 个字不分享
24 }
25 document.onclick=function (){ //点击页面
26     share_button.style.display='none'; //分享按钮消失
27 }
28 text.onclick=function (){ //点击文本时阻止 document.onclick
29     var ev=ev||window.event;
30     ev.cacelBubble=true;
31 }
32 share_button.onclick=function (){ //http 请求分享到新浪微博
33     window.location.href="http://service.weibo.com/share/share.php?title="+
34     select()+"&url="+window.location.href;}}
```

注意：随着社交网络的不断完善，目前已有很多集成式分享的解决方案，如 bshare、jiathis 等。在使用时只需向页面引入一行 JavaScript 代码即可，非常方便，建议在做集成式分享时采取分享工具的方案。

本节的实例可以拓展到划词翻译。在代码方面，划词翻译与选中分享大致相同，只是在 JavaScript 方面可通过 AJAX 请求数据然后加载至悬浮层这部分代码有所不同。

3.12 链接百叶窗效果

链接百叶窗效果如图 3.24 所示，左图和右图是链接的两种状态，这两种状态可以像百叶窗一般切换，中图是切换过程中的截图。使用该效果既可以丰富网页的效果，又可以在一定的区域内展示更多的链接，是链接布局的不错选择。

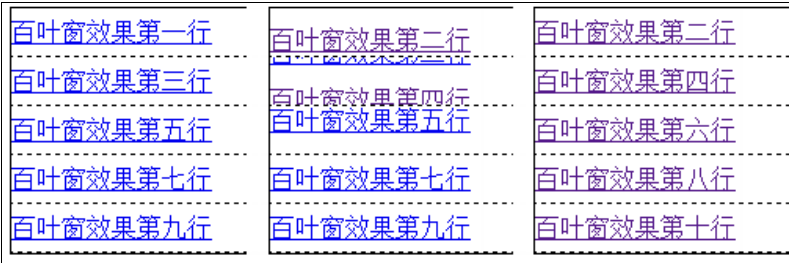


图 3.24 链接百叶窗效果

HTML 部分的代码的结构为：ul>li>div>a， “>” 符号为包含关系。在 CSS 布局方面，每个 li 的大小固定、溢出隐藏、采用相对定位的方式。而 li 中的 div 绝对定位、高度为 li 的两倍，这导致 div 只有一部分显示出来。通过改变 div 的 top 值，可以改变 li 中可以看到的内容。对每个 li 中的 div，可以采用元素运动的方式平滑的切换 top 属性，同时让每个 div 按从上到下的顺序依次运动，就能得到平滑的、有错落感的百叶窗动画效果。CSS 部分的代码如下：

```
01  *{margin:0;padding:0;}
02  #blinds{width:300px;height:auto;border:1px solid;margin:20px auto;}
03  #blinds>li{
04    list-style:none;width:100%;height:30px;overflow:hidden;
05    position:relative;border-bottom:1px dashed;line-height:30px;}
06  #blinds>li>div{position:absolute;top:-30px;}
07  #blinds>li>div>a{height:30px;}
```

JavaScript 部分是通过改变 CSS 第 6 行中的 top 值实现动画的。下面的这段 JavaScript 代码是实现此效果的一种方案，其中运用了运动框架使元素达到运动的效果。代码中总共有两个定时器，外层定时器控制链接切换的频率，内层的定时器则控制切换的次序从而产生错落感。

```
01  <script type="text/javascript" src="zQuery.js"></script>
02  <script type="text/javascript">
03  window.onload =function (){
04    var blinds=document.getElementById("blinds");           //获取 ul
05    var btn=true;                                           //定时器、切换判断变量
06    show(blinds);
07    function show(obj){                                     //展示切换函数
08      var timer=setInterval(function(){
09        var i=0;
10        var blinds_div=obj.getElementsByTagName('div');     //获取 div
11        var t=setInterval(function (){                     //内层定时器
12          if (i<blinds_div.length)                         //按次序切换过程
13          {
14            console.log(i);
15            //运动函数，通过切换判断变量决定向上/向下切换
16            move(blinds_div[i],{top:btn?0:-30});
17            i++;                                              //准备切换下一个元素
18          }else{                                             //本次切换完成
19            btn=!btn;    //切换方向变量取反（如本次向下切换，下次会向上切换）
20            clearInterval(t);                                //清空当前的定时器
21          }
22        },80);
23      },4000);
24    }
25  </script>
```

3.13 iPhone 开关

iPhone 界面模拟是 Web 应用开发中的一个分支。本节的实例是使用 CSS 创建 iPhone 开关的效果（如图 3.25 所示），并不涉及 JavaScript 的内容，只是用 CSS 的过渡效果模拟动画的效果（如点击开关切换在实例中只是用 hover 模拟样式的改变）。



图 3.25 iPhone 开关

文字的外层是一个无序列表，每一项都是一个列表项。开关分为 2 层，外层为轮廓（是一个 span），第 2 层有一层是轮框宽度 2 倍的 span 并使用 after 伪对象添加中部的圆形小块，第 2 层上方有 2 个部分，分别是左边的开状态和右边的关状态符号 1 和 0，具体结构如下：

```
01 <ul><p>设置</p>
02 <li>
03 <span class="iphoneButton"><span class="button buttonOn">
04 <span class="on"></span><span class="off"></span>
05 </span></span>
06 飞行模式
07 </li>
```

对列表样式进行设置，主要是设置整体与各个列表项的布局关系，由于之后的按钮内层使用定位，故从结构的外层开始，依次设置定位方式避免混乱的方式。

```
01 ul{width:230px;padding:0;
02 border:1px solid;border-radius:5px; /*外部圆角框*/
03 position:absolute;left:22px;top:100px;
04 }
05 ul p{font-size:15px;text-align:center;margin:5px;} /*标题的设置*/
06 li{list-style:none;
07 width:90%;height:30px;line-height:30px;padding-left:10%; /*行高设置使文字竖直居中*/
08 border:1px solid;border-bottom:none;
09 position:relative;
10 }
```

对按钮部分的做外层的设置：

```

01 .iphoneButton{margin:2%;width:30%;height:65%;
02 border:1px solid;border-radius:15px 15px;
03 overflow:hidden;display:block;
04 position:absolute;right:0;
05 }

```

对按钮内层部分的设置，并在内层的中部使用 `after` 创建一个圆圈伪对象。由于背景有两种颜色，前半部分为蓝色，后半部分为灰色，所以背景颜色使用渐变来设置（渐变为 W3C 标准）。第 8~9 行设置 CSS 3 的过渡效果，使得开关的切换缓缓进行。

```

01 .button{width:180%;height:100%;padding:0;
02 position:absolute;
03 line-height:100%;
04 font-weight:bold;
05 background:linear-gradient(to right,#1E90FF 0%,#1E90FF 50%,
06 #EDEDED 50%,#EDEDED 100%); /*使用渐变可以减少一个按钮*/
07 border-radius:15px 15px;
08 transition:all .5s ease;
09 -webkit-transition:all .5s ease;
10 }
11 .button:after{content:"";
12 width:15%;height:96%;
13 border:1px solid;border-radius:100%; /*开关中央的圆形可以使用伪对象实现*/
14 position:absolute;left:40%;
15 background:#FFFFFF;
16 }

```

设置 1 和 0，分别使用两个定位的 `span` 实现，前者使用只有边框的 `span`，后者使用完全圆角的 `span` 制作圆形。

```

01 .on{width:0;height:50%;margin-top:3%;margin-left:22%;
02 border:1px solid #FFFFFF;
03 position:absolute;
04 }
05 .off{width:8%;height:50%;margin-top:2%;margin-left:72%;
06 border:2px solid #A1A1A1;border-radius:100%;
07 position:absolute;
08 }

```

最后设置开关的打开状态、关闭状态以及鼠标悬浮切换：

```

01 .button:hover{left:-73%;}
02 .buttonOn{left:-73%;}
03 .buttonOff{left:0;}

```

3.14 按钮式单选框与复选框

因用户使用鼠标比使用键盘更加普遍，故网页设计中经常使用单选框与复选框方便用

户 check，在 HTML 中通过 radio 和 checkbox 两种类型的输入框实现。原生的单选框为小圆点样式，原生的复选框为 √ 和小方块，本节的内容是自定义这些元素的样式，不使用原生的样式，同时并不影响表单的正常填充，效果图如图 3.26 所示。需要用到小部分的 JavaScript，本实例使用的是 jQuery。

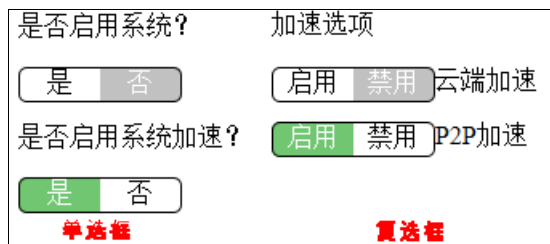


图 3.26 自定义选框

单选框的实现思路是在 HTML 部分添加两个 label 标签，分别指向两个单选选项，在 CSS 部分设置 label 标签的开启和禁用两种样式，当用户切换点击 label 标签的同时会自动触发单选选项的选中，在 JavaScript 切换 label 标签的 class 实现样式的改变。最后把原生的 input 框设置为不显示。

复选框的实现思路与单选框类似，同样是两个 label 标签在不同样式之间的切换。不同之处是用户点击的同时，通过 JavaScript 改变复选框的 check 属性值，而不是自动触发复选框选中（由于单选框必选一项，不必考虑解除选中，而复选框可以不选择，自动触发容易，自动解除选中状态却无法实现）。

//单选框结构

```
<p>
  是否启用系统?
  <input type="radio" id="radio1" name="field" checked />
  <input type="radio" id="radio2" name="field" />
  <div>
    <label for="radio1" class="radioEnable enable"><span>是</span></label>
    <label for="radio2" class="radioDisable "><span>否</span></label>
  </div>
</p>
```

//复选框结构

```
<p>
  <input type="checkbox" id="radio5" checked/>
  <label class="radioEnable enable cbEn" ><span>启用</span></label>
  <label class="radioDisable cbDis"><span>禁用</span></label>
  云端加速
</p>
```

对启用状态的按钮和禁用转台的按钮设置样式如下：

```
01 .radioEnable{ /*开状态设置*/
02   width:50px;height:20px;
03   border-radius:5px 0 0 5px/5px 0 0 5px ;
```

```

04 float:left;text-align:center;
05 border-top:1px solid black;           /*开、关状态不同之处主要在边框上*/
06 border-left:1px solid black;
07 border-bottom:1px solid black;
08 }
09 .radioDisable{                       /*关状态设置*/
10 width:50px;height:20px;
11 border-radius:0 5px 5px 0/0 5px 5px 0 ;
12 float:left;text-align:center;
13 border-top:1px solid black;
14 border-right:1px solid black;
15 border-bottom:1px solid black;
16 }

```

对按钮选中时的颜色及样式进行设置，使得启用状态选中的颜色为绿色，禁用状态选中的样式为灰色。

```

01 .radioEnable.enable{
02   background:#71C671;color:#fff;
03 }
04 .radioDisable.enable{
05   background:#C1C1C1;color:#fff;
06 }

```

最后把 input 标签默认设为不显示。

```
01 input{display:none;}
```

CSS 部分的两种样式做完后，JavaScript 部分的 class 切换就简单了。当按钮被单击后，切换按钮的样式为选中的样式，同时将对立的状态转换为初始样式，同时将选项选中。为了使上面的讲述明确，附上一段 jQuery 解决方案以明确 class 的切换实现。

```

01 $(function(){
02   $(".radioEnable").click(function(){           //单选启用按钮被单击
03     $(this).parent('div').children(".radioDisable").removeClass('enable'); //禁用按钮去除选中状态
04     $(this).addClass('enable');                 //启用按钮选中
05   });
06   $(".radioDisable").click(function(){          //单选禁用按钮被单击
07     $(this).parent('div').children(".radioEnable").removeClass('enable'); //启用按钮去除选中状态
08     $(this).addClass('enable');                 //禁用按钮选中
09   });
10   $(".cbEn").click(function(){                  //复选启用按钮被单击
11     $(this).parent('p').children(".radioDisable").removeClass('enable'); //禁用按钮去除选中状态
12     $(this).addClass('enable');                 //启用按钮选中
13     $(this).parent('p').children('input').prop('checked',true);           //input 属性值选中
14   });
15   $(".cbDis").click(function(){                 //复选禁用按钮被单击
16     $(this).parent('p').children(".radioEnable").removeClass('enable');   //启用按钮去除选中状态
17     $(this).addClass('enable');                 //禁用按钮选中
18     $(this).parent('p').children('input').prop('checked',false);           //input 属性值去除选中

```

```
19  });
20  });
```

3.15 自定义播放器

随着 HTML 5 的不断普及,网页支持视频的方式已经不再局限于 Flash Player 或者使用其他插件了,使用 HTML 5 的一个 video 标签即可搞定。YouTube 和优酷等许多网站已经率先支持了 video 标签,Adobe 公司对 Flash Player 的更新也日趋缓慢,这也从侧面映射出 HTML 5 必然是大势所趋,因为它带来的不仅仅是标签和属性的增加,同时还有多种媒体的支持和 canvas 画布等。

在 video 标签中,可以使用 controls="controls"使得视频上自动带有一个控制模块,该模块在 IE 10 和 IE 11 浏览器上表现良好,与 Windows 8、Windows 8.1 和 WP 手机系统相得益彰,在其余的浏览器上却不是特别美观。但如果不使用默认的控制模块,完全可以通过 CSS 和 JavaScript 改造 button、a、div、span 等几乎所有 HTML 元素成为一个美观的控制模块。本节实现的自定义播放器样式如图 3.27 所示。



图 3.27 自定义播放控件

由于 HTML 5 对视频的控制给 JavaScript 很大的方面,所以控制方面很容易实现。另外,控件的所有按钮完全是 HTML 的各种标签,在美化方面做的工作其实就是美化和定位而已。

在 HTML 文档方面,结构如下,第 1 行为 video 标签,src 属性值为视频文件路径。第 2~13 行为控件区,其中包含第 3 行播放进度条、第 4 行当前时间与总时间提示、第 5~12 行中播放与暂停、快进快退、静音与取消静音、音量控制条以及全屏按钮,音量控制条采用 HTML 5 中的滑块标签。

```
01 <video src="mpeg/love.mp4" class="player">不支持 HTML5 提示 </video>
02 <div class="playerBar">
```

```
03 <div id="progressbar"><div id='progress'></div></div>
04 <p class="time"></p>
05 <span class="play"></span>
06 <span class="pause"></span>
07 <span class="go10"></span>
08 <span class="back10"></span>
09 <input type="range" class="vol"/>
10 <span class="muted1"></span>
11 <span class="muted0"></span>
12 <span class="full"></span>
13 </div>
```

CSS 部分主要工作为布局定位和设置按钮的背景图片，对于播放进度条部分，照搬变换与动画一章中的进度条部分的结构、CSS 和 JavaScript 函数稍加修改即可，不再赘述，可参照实例。

[illegible]

```

30 background:url("img/9.png");background-size:32px 32px;
31 width:32px;height:32px;
32 }
33 .muted0{                                /*音量*/
34 bottom:2%;right:30%;
35 background:url("img/14.png");background-size:32px 32px;
36 width:32px;height:32px;
37 }
38 .muted1{                                /*静音与音量重叠*/
39 bottom:5%;right:30%;
40 background:url("img/15.png");background-size:32px 32px;
41 width:32px;height:32px;
42 }
43 .full{                                  /*全屏*/
44 bottom:5%;right:1%;
45 background:url("img/16.png");background-size:32px 32px;
46 width:32px;height:32px;
47 }
48 .vol{                                  /*音量滑块*/
49 width:20%;
50 bottom:30%;right:8%;
51 color:#ccc;
52 }
53 .time{                                  /*时间提示*/
54 position:absolute;top:15%;right:0;
55 font-size:12px;
56 }

```

既然浏览器能够支持 HTML 5，那么背景大小设置也必然能够支持。CSS 代码中的定位多使用百分比以加强不同状态下的兼容性。

JavaScript 部分的工作有点击按钮播放器响应操作、动态设置当前的时间以及播放进度以及实现滑块控制音量等功能。本实例为了方便起见，采用 jQuery 实现功能，具体的实现过程如下代码所示。

```

01 $(document).ready(function(){
02     var video = $('video');                //获取 video 标签
03     $(".pause").css('display','none');      //页面载入完成将暂停键隐藏、只显示播放键
04     $(".play").click(function(){           //播放键被点击
05         video[0].play();                   //响应播放
06         $(".play").css('display','none');   //播放键隐藏
07         $(".pause").css('display','block'); //切换为暂停键
08         var all=video[0].duration;          //获取视频总秒数
09         var allhour=parseInt(all/3600);     //计算小时
10         var allminute=parseInt((all-allhour*3600)/60); //计算分钟

```



```

11     var allseconds=parseInt(all-allhour*3600-allminute*60);//计算秒
12     var t=setInterval(function(){                                //设置一个间隔 1s 的定时器
13         var now=video[0].currentTime;                          //每秒获取一次当前播放时间
14         if (now==all)                                           //播放完成清空定时器
15         {
16             clearInterval(t);
17         }
18         var cent=now/all*100;                                    //计算播放进度
19         progressfn(cent);                                       //传入进度条变化函数
20         var hour=parseInt(now/3600);                             //计算当前小时
21         var minute=parseInt((now-hour*3600)/60);               //计算当前分钟
22         var seconds=parseInt(now-hour*3600-minute*60);        //计算当前秒
23         $(".time").html(hour+": "+minute+": "+seconds+'/'
24             +allhour+": "+allminute+": "+allseconds);
25                                     //将时间信息写入到网页中
26     },1000);
27 });
28 $(".pause").click(function(){                                //暂停键被点击
29     video[0].pause();                                         //响应暂停
30     $(".pause").css('display','none');                       //暂停键隐藏
31     $(".play").css('display','block');                       //切换为播放键
32 });
33 $(".go10").click(function(){ video[0].currentTime+=2; });/    //快进 2s 操作
34 $(".back10").click(function(){ video[0].currentTime-=2; }); //快退 2s 操作
35 $(".muted1").click(function(){                               //静音键被点击
36     video[0].muted=false;                                    //响应静音
37     $(".muted1").css('display','none');                      //静音键隐藏
38     $(".muted0").css('display','block');                    //音量键显示
39 });
40 $(".muted0").click(function(){                               //音量键被点击
41     video[0].muted=true;                                     //响应静音
42     $(".muted0").css('display','none');                      //音量键隐藏
43     $(".muted1").css('display','block');                    //静音键显示
44 });
45 $(".full").click(function(){                                //全屏键被点击（无 IE）
46     video[0].mozRequestFullScreen();                        //Firefox 浏览器
47     video[0].webkitEnterFullscreen();                      //Webkit 类型的浏览器
48 });
49 $(".vol").click(function(){                                //音量滑块控件被点击
50     video[0].volume=$(".vol").val()/100;                  //获取滑块的值并转换为播放音量的值并设置
51 });
52 });
53 function progressfn(cent)                                    //进度条改变函数

```

```

54 {
55   var progressbar=document.getElementById("progressbar");
56   var progress=document.getElementById("progress");
57   progress.style.width=cent+"%";
58 }

```

注意：原生的 HTML 5 的 video 标签在支持的视频格式方面远远不如 Adobe Flash Player，只支持 mpeg4（mp4）、ogg 和 webm 格式，可采用第三方 JavaScript 库辅助支持更多的视频格式。

3.16 文字变链接

如果想在一篇文章中的某些关键字上添加超链接，但是同时不影响文章的原来面目，那么本节的效果是非常有效的。如图 3.28 所示，初始状态下的文字样式在鼠标指向时变为了链接，从而可以让用户链接到相应的页面上。

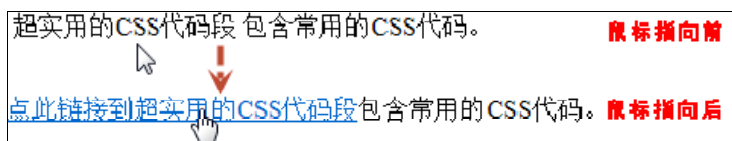


图 3.28 文字变链接

该效果可以使用简单的两行 CSS 代码完成，文字和链接在网页中实际都存在，只是在鼠标不指向文字时链接不显示，指向文字时文字隐藏同时链接展示出来。以下是 HTML 和 CSS 代码。

```

//HTML 代码
01 <div>
02   <span>超实用的 CSS 代码段</span>
03   <a href="">点此链接到超实用的 CSS 代码段</a>包含常用的 CSS 代码。
04 </div>
// CSS 代码
01 a, div:hover span { display: none; }           //链接隐藏、文字显示
02 div:hover a { display: inline-block; }         //链接显示、文字隐藏

```

3.17 根据文件格式设置链接图标

a 标签在网页中是经常被使用的，CSS 可以实现筛选 a 标签属性，由此可以根据文件的格式设置不同的链接图标来增强用户体验，在每种格式对应好一种图标之后，用户对自己将要访问或者下载的文件一目了然，如图 3.29 所示。



图 3.29 链接不同图标效果图

以下的实现有 3 种代码形式：第一种是匹配 href 属性以关键字开头的标签；第二种是匹配 href 属性以关键字（一般是文件格式）结束的标签；最后一种可以匹配到 href 属性中包含某关键词的标签。CSS 匹配标签后可以设置标签的背景图片作为链接的图标。

```

01 a[href^="http:"] {
02     display:inline-block;                      /* 图片显示方式 */
03     padding-left:20px;                          /* 设置左 padding */
04     background:transparent url("img/url.gif") center left no-repeat; /* 图片居左显示 */
05 }
06 a[href$=".pdf"] {
07     display:inline-block;
08     padding-left:20px;
09     line-height:18px;                          /* 可以设置行高 */
10     background:transparent url("img/pdf.gif") center left no-repeat;
11 }
12 a[href *="username"] {
13     padding-left: 20px;
14     background: url("img/star.png") no-repeat left;
15 }

```

第 1 行的选择表达式可使 CSS 找到所有链接 href 属性以“http:”开头的链接，第 6 行的选择表达式可使 CSS 找到所有链接 href 属性以“pdf”结尾的链接，同理，第 12 行选择的是 href 属性包含“username”的所有链接，括号内的代码分别设置链接样式。

注意：源代码中附带了常用的基本文件图标的集合，便于扩展使用。

3.18 链接标签“a”的顺序

a 标签有以下几个属性。

- active: 设置当链接处于激活状态时 a 标签的样式。
- hover: 设置当用户将鼠标指针悬停在链接上时 a 标签的样式。
- link: 设置当链接最近没有访问过时 a 标签的样式。
- visited: 设置当链接最近访问过时 a 标签的样式。

同时需要注意的是 a 标签的 link、visited、hover、active 这 4 个属性是有一定顺序关系的。其中 a:hover 必须被置于 a:link 和 a:visited 之后，才是有效的。a:active 必须被置于 a:hover 之后才有效。具体实现可见如下代码：

```

1 //CSS 代码
2 a:link {

```

```
3   color: #0000FF;
4   text-decoration: underline;
5   font-weight: normal;
6   font-style: normal;
7 }
8 a:visited {
9   color: #3399FF;
10  text-decoration: underline;
11  background-color: #FFFFFF;
12  font-weight: normal;
13  font-style: italic;
14 }
15 a:hover {
16   color: #0000FF;
17   text-decoration: underline;
18   background-color: #FFFF00;
19   font-weight: bold;
20   font-style: normal;
21 }
22 a:active {
23   color: #FF0000;
24   text-decoration: none;
25   background-color: #CCCCCC;
26   font-weight: bold;
27   font-style: normal;
28 }
```

以上代码中的样式表在第 1 行、第 7 行、第 14 行、第 21 行分别定义了链接、已访问过的链接、鼠标停在链接上方时、鼠标单击链接时的样式。此处定义这 4 种类型的样式表时需要注意的是，必须按以上顺序定义样式，否则链接显示的效果与预期可能不一致。

a 标签的 link、visited、hover、active 这 4 种类型的显示效果如图 3.30 所示。

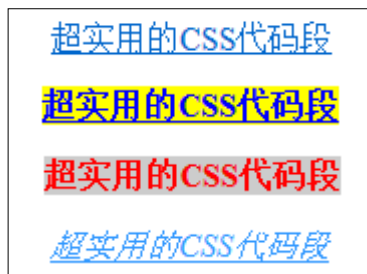


图 3.30 a 标签显示效果

图 3.30 从上至下依次是 link、visited、hover、active 的显示效果。

a:link 类型出现的时间为最早，可应用于所有的链接。a:visited 类型次之，它将覆盖任何链接的 a:link 样式，即如果 a:link 类型紧跟着的是 a:visited，a:link 可能会覆盖 a:visited 类型定义的样式而呈现 a:visited 定义的样式。再次是 a:hover 类型，此类型只应用于访问鼠

标下的链接。`a:visited` 类型中包含指定所有与 `a:hover` 相同属性的规则，否则，当访问者的鼠标通过一个访问链接时，没有被 `a:visited` 类型取代的任何 `a:hover` 类型属性将会继续再现。最后是 `a:active`，当链接被鼠标单击时，`a:active` 样式可以取代所有其他 3 种类型所定义的风格。

在 CSS 代码中的类型顺序确定了每一种类型如何取代其他的类型，即更多的类型可以应用到特定的元素上。正常情况下，`a:hover` 类型处于 `a:link` 和 `a:visited` 类型之后，所以 `hover` 状态的类型可以应用于常规和访问的链接。但是，它也并非必须遵循这一方式，根据需要，可以改变类型顺序实现不同的效果，也可以只定义这 4 种类型其中的一种或多种。

一个便于记忆的缩写是“LVHA”，即 `a` 标签的 4 种伪类的首字母的组合。



第 4 章 背景和颜色

背景和颜色是为丰富效果而生的属性，在任何场合下，都是不可或缺的。在网页设计中背景和颜色通常是美化方面的主力，真正独立的实例不多。

本章主要涉及的知识点有：

- 颜色和渐变的基础与实例
- 高光效果、页面顶部阴影
- 多背景
- 全屏背景
- 斑马线背景
- 棋盘背景
- 易拉罐效果

4.1 颜色和渐变的基础与实例

4.1.1 颜色

在 CSS 1 和 CSS 2 中，颜色的值为部分颜色名称、十六进制和 RGB 色。CSS 3 新增了 RGBA、HSL 和 HSLA，融合了饱和度、透明度等概念，使得网页颜色更加丰富。

- **RGBA**

RGBA 在 RGB 的基础上增加了透明度（alpha）。R、G、B 的取值与 RGB 相同，可以为 0~255 的数值或百分比 0~100%，透明度的值与 CSS 3 中 opacity 的值相同，为 0~1 之间的数。

- **HSL**

HSL 色彩模式是工业界的一种颜色标准，是通过色调 H、饱和度 S、亮度 L 3 个颜色通道的变化以及它们的叠加来得到各式各样的颜色，这一标准几乎包括了人类视力能力所能感知的所有颜色。于是 HSL 的参数有 3 个，H 的值为 0~360，0~119（或 360）表示红色、120~239 表示绿色、240~359 表示蓝色。S 和 L 取值为 0~100%。

- **HSLA**

HSLA 即在 HSL 的基础上增加了透明度。A 的取值在 0~1 之间。

注意：所有的颜色建立在美学基础之上，CSS 中的参数与美学上的颜色参数是一致的。

4.1.2 渐变简述

在本书的诸多实例中，有很多关于 CSS 3 渐变的应用，在实例中对渐变部分的实现并无详细的说明。CSS 3 渐变是一个很强大的功能，但是越强大的东西往往越复杂。从 Webkit 率先实现渐变，到各大浏览器先后实现带有自己品牌（CSS 前缀）的渐变，再到现在各大主流浏览器向无前缀的 W3C 标准改进并得到良好的支持，这一路已走了很长时间。

由于国内浏览器的发展形势，尤其是性能差的低版本 IE 浏览器在国内的比重依然很大，为了兼容，国内的 CSS 3 一直没有快速的发展。每一个网站开发人员不得不考虑 IE 低版本、各种前缀以及 W3C 标准，这使得在国内应用 CSS 3 新特性变得困难且头痛。对于渐变来说，在国内没有大面积应用的时候，国外的这一技术却正在向改进发展。这既是好的，因为国内的开发人员可以直接使用 W3C 标准开发，少走弯路；却又是坏的，因为浏览器的市场占有率，导致新旧交替的属性都要考虑。

由于上述问题，本书中很多实例没有当面讲清渐变的实现，但为了使读者更加了解渐变的使用，这里有必要单独列出一节来细致地讲解 CSS 3 渐变。本节的内容将更加青睐与 W3C 标准和新旧交替。

注意：渐变的语法有两种，一种是渐变类型写在参数内的方式（如渐变（线性，参数）），另一种是渐变类型写在参数外的方式（如径向渐变（参数））。本节中采用后者，因为后者更方便，不同浏览器之间也更统一。IE 的 filter 滤镜在本节中没有涉及。

4.1.3 带前缀的渐变

各大浏览器的前缀分别为 -webkit-、-moz-、-o- 以及 -ms-。

（1）线性渐变

基本语法为：前缀-linear-gradient(渐变线起点位置或使用角度确定渐变线、开始颜色、[中间多个颜色[及位置]]、结束颜色)。“[]”代表可选。

渐变线的确定可以采用关键字或者角度。关键字有 top、bottom、left、right、top left、top right、bottom left 和 bottom right。使用关键字时，渐变线为从关键字开始到关键词的对面结束，如 top 的意义为渐变线从上到下。使用角度时，渐变线为指向右侧的渐变线沿逆时针方向旋转一定的度数。如 30deg 的意义如图 4.1 所示。

颜色值可有多种多样，可以使用关键字、16 进制、rgba 等，浏览器支持就好，还可以使用透明（transparent）。中间颜色可选，作用是在开始颜色和结束颜色之间设置颜色停靠点（color-stop、色标）。如：

```
-webkit-linear-gradient(top,red,yellow 25%,green 50%,blue 75%,purple);
```

代码的效果如图 4.2 所示。

（2）径向渐变

与线性渐变相比，径向渐变的参数更为复杂。基本语法为：前缀-radial-gradient([径向渐变的圆心位置],[渐变的形状][渐变的尺寸],开始颜色,[中间颜色],结束颜色)，“[]”表示参数可选。

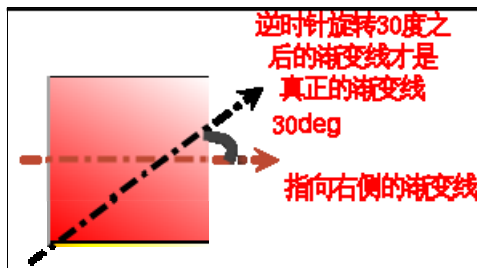


图 4.1 使用角度确定渐变线

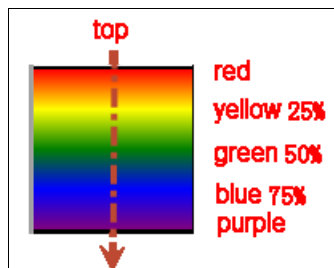


图 4.2 效果展示

径向渐变的圆心位置有 3 类值可以选择：一类是长度值，可为负值，如 `-ms-radial-gradient(-30px 60px,red,white)`；一类是百分比，可为负值，如 `-ms-radial-gradient(30%-60%,red,white)`；另外一类为关键字，如 `-ms-radial-gradient(top,red,white)`。此参数不写时浏览器使用默认参数 `center`。

径向渐变的形状有两个值可选：`circle` 为圆形，`ellipse` 为椭圆形，圆形是椭圆形的特殊情况（长轴与短轴相等）。该参数不写时，通过大小值确定形状，大小参数也不写时，形状为椭圆形。

渐变的大小有 3 类值可以选择，一类是长度值，如 `-ms-radial-gradient(center,30px 30px,red,white)`；一类是百分比，如 `-ms-radial-gradient(center,30% 30%,red,white)`；另一类是关键字。`closest-side`：指定径向渐变的半径长度为从圆心到离圆心最近的边、`closest-corner`：指定径向渐变的半径长度为从圆心到离圆心最近的角、`farthest-side`：指定径向渐变的半径长度为从圆心到离圆心最远的边、`farthest-corner`：指定径向渐变的半径长度为从圆心到离圆心最远的角，默认值为 `farthest-corner`。使用四种关键字的代码如下，效果如图 4.3 所示。

```
background-image:-ms-radial-gradient(30px 40px ,circle closest-side ,red,white);
background-image:-ms-radial-gradient(30px 40px ,circle closest-corner ,red,white);
background-image:-ms-radial-gradient(30px 40px ,circle farthest-side ,red,white);
background-image:-ms-radial-gradient(30px 40px ,circle farthest-corner ,red,white);
```

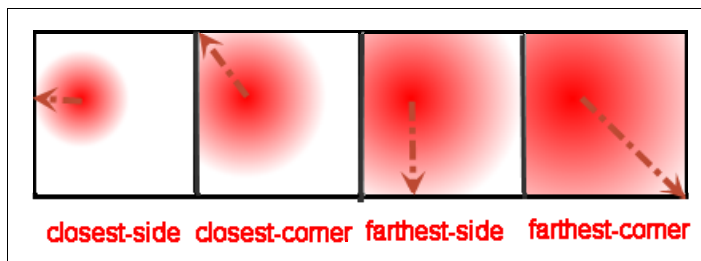


图 4.3 大小关键词

注意：在确定形状和大小时，通常在不指定形状时使用长度值或百分比（如 30% 40%）确定水平半径和竖直半径，这时形状便会随之确定。或者使用形状与大小关键字的组合（如 `circle closest-side` 或 `closest-side`），使用前者能够更加灵活地控制渐变区域的大小。

颜色参数的设置及意义与线性渐变相同。

注意：使用径向渐变可以轻易实现一个立体小球效果。

4.1.4 W3C 标准渐变（不带前缀）

各大主流浏览器均已支持 W3C 标准语法，标准语法与带前缀语法有细微的不同。

（1）线性渐变

语法：linear-gradient(to 渐变线终点位置或使用角度确定渐变线、开始颜色、[中间多个颜色[及位置]]、结束颜色)。“[]”代表可选。使用角度时，渐变线为指向顶部的渐变线沿顺时针方向旋转一定的度数（更改后更加符合 SVG 的特点）。如 30deg 的意义如图 4.4 所示。用法如下：

```
linear-gradient(to bottom,red,white);
linear-gradient(30deg,red,yellow,white);
```

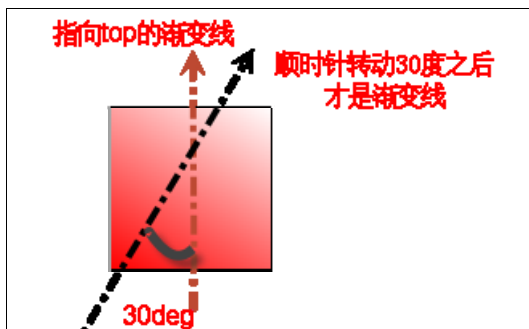


图 4.4 W3C 标准的渐变线确定

（2）径向渐变

语法：radial-gradient([渐变的形状][渐变的大小][at 径向渐变的圆心位置],开始颜色,[中间颜色],结束颜色)， “[] ” 表示参数可选。用法如下：

```
radial-gradient(30px 40px at 40px 40px ,red,white);
```

注意：对于新手来说，简单地看过上述的文字不一定能很好地理解 CSS 3 渐变的使用，建议新手把可能的各种组合自行列出查看效果以便更快地理解渐变。

可以看到，W3C 标准与带前缀版本并不是完全一致的，在不明确的时候不能乱用前缀。为了使代码的兼容性达到最好。建议把前缀版本写在前，W3C 标准写在后，以适应各种浏览器的需要。

4.1.5 重复渐变

在没有重复渐变之前，需要使用 background-size 和 background-repeat 制作重复渐变的效果，当对渐变效果来讲，这是非常不方便的，好在 CSS 3 中补充了重复渐变的属性，通过 repeating-linear-gradient 和 repeating-radial-gradient 替换 linear-gradient 和 radial-gradient 就能实现重复渐变。

注意：动画边框实例中不改变背景大小的另一种方式为这里所说的重复渐变。

4.2 高光效果

在文字与字体和按钮与链接等章节中曾经运用过高光效果，高光效果往往与阴影效果、渐变效果和描边同时使用以增强立体感，图 4.5 为高光效果的效果图。这种效果常常用在按钮的制作上。



图 4.5 高光效果

图中 div 的描边和阴影效果为边框属性和外边框阴影，渐变则是颜色渐变基础应用，高光效果为内阴影，该阴影白色半透明。CSS 代码如下。

```
01 div{
02   width:200px;height:50px;margin:30px auto;padding-top:10px;
03   border:1px solid #0059ff;                                /*描边效果*/
04   border-radius:10px;                                       /*圆角效果*/
05   background:-webkit-linear-gradient(top,#589dfd 0%,#488bf7 50%,#3a7af2 100%);
06   background:-moz-linear-gradient(top,#589dfd 0%,#488bf7 50%,#3a7af2 100%);
07   background:-o-linear-gradient(top,#589dfd 0%,#488bf7 50%,#3a7af2 100%); /*渐变*/
08   background:-ms-linear-gradient(top,#589dfd 0%,#488bf7 50%,#3a7af2 100%);
09   background:linear-gradient(to bottom ,#589dfd 0%,#488bf7 50%,#3a7af2 100%),
      #0059ff;
10   box-shadow:0 0 5px rgba(255,255,255,0.7) inset,1px 1px 3px 1px #ccc;
11 }
```

其中第 9 行使用了 CSS 3 对 background 的新增属性，通常把 background 属性的值取为背景图片和背景色，用逗号隔开，当背景图片不可用时，背景将使用纯色。

注意：可以使用网络上的按钮生成工具方便快捷地生成效果，如果读者对 Photoshop 有所了解会发现效果的实现与 Photoshop 中的常见实现原理相同，网页很多应用均为多层的叠加效果。

4.3 多背景

CSS 3 中加强了背景的设置，其中多背景解决了以前只能使用一张图片作为背景的问题，可以使用多张图片拼接一张背景。图 4.6 为多背景实例效果。

代码为 CSS 2 的升级，第一行使用 background-image 指定背景图片，多个背景图片之间使用逗号隔开，同时第 3 行和第 4 行对背景重复属性和背景图片位置属性的设置也需要设置多次。

```

01 background-image:url("img/1.jpg"),url("img/2.jpg"),url("img/3.jpg"),
02                      url("img/4.gif"),url("img/5.jpg");
03 background-repeat:no-repeat,no-repeat,no-repeat,no-repeat,no-repeat;
04 background-position:top left,top right,bottom left,bottom right,center center;

```

在不指定背景位置之前，所有的背景图片是重叠在一起的，根据这一点可以制作一个使用透明图片合成的一个背景图（利用图片的透明效果合成，不透明图片会互相覆盖，覆盖使得属性中的最后一张图片在最上层），如图 4.7 所示，总效果为图片 1 和图片 2 合成得来。

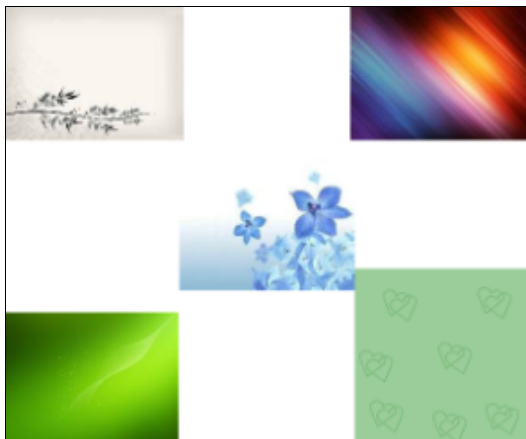


图 4.6 多背景



图 4.7 多图片合成

4.4 全屏背景

CSS 3 加强了对背景的控制，其中对背景大小的设置常用，在使用 CSS 控制背景大小时，可以使用具体的像素值，也可以使用百分比（占父元素的百分比），还可以使用 **cover** 和 **contain** 两个关键词，这提高了图片的重用率。**cover** 关键词的含义是图片的大小自动适应区域大小，而 **contain** 的含义为自动根据区域的大小适应图片的大小，所以使用 **cover** 关键词图片有时不能完全显示在区域内（超出部分会覆盖掉），而使用 **contain** 图片的所有细节都会显示出来。对于一个全屏背景来说，使用的关键词为 **cover**。代码如下：

```

01 html{                                     /*在整个 html 上加样式*/
02   background:url("img/1.jpg") no-repeat center center;
03   background-size:cover;
04   min-height:100%;                         /*设置 html 的最小高度*/
05   min-width:100%;
06 }
07 body{
08   min-height:100%;                         /*设置 body 的最小高度*/
09 }

```

在整个 HTML 上添加背景，这将比在 **body** 上设置背景更加有效。关键之处在于设置

HTML 的最小高度为 100%，之后使用 `cover` 关键词使得背景自适应 HTML 的大小。第 1 行中的 `center` 可使窗口很小的时候使用背景的中部部分（一般图片的重点在中部位置，视情况而定）。这段代码可以在窗口大小调整时自动调整背景，保持背景的全屏显示。

注意：从实例的效果看，代码中的第 5~9 行并不是必要的，但为了防止特殊情况的发生，可以将其写入代码中。

4.5 斑马线背景

有些网页设计人员在美化网页背景时喜欢设置成斑马线样式的背景。事实上在国内的诸多网页中，这一效果应用并不广泛，这也与个人的品位有关，有的人喜欢，有的人却未必喜欢这种背景，然而这并不重要，之所以把本节内容呈现出来，是因为许多其他的 CSS 效果与本节有联系，重点是呈现这种实现思路。

在 CSS 3 之前，背景的实现通常是通过背景图片平铺的方式。本节是使用 CSS 3 中的渐变实现条纹，这种思想在第 2 章 CSS 动画边框中已经有所展现。本节代码的兼容性为 IE10+、Chrome 浏览器、Firefox 浏览器、Opera 浏览器或者 Safari 浏览器。在 IE 9 以下版本浏览器上，恐怕要通过背景图片或者 `filter` 滤镜来实现了。本节的效果图如图 4.8 所示。

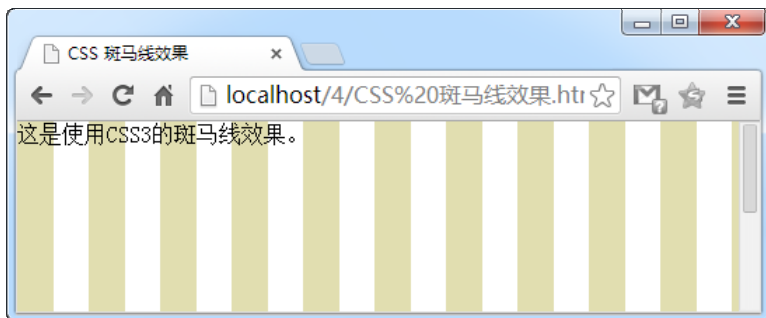


图 4.8 CSS 斑马线背景

在 HTML 文档里不需要写什么东西，但是必须保证 `body` 区域不为空，然后直接为 `body` 区定义背景。下面是整个 CSS 代码：

```
01  *{margin:0;padding:0;}
02  body{
03    background-image:-webkit-linear-gradient(0deg,#E1DEB0 50%,transparent
04    50%,transparent);                                /*渐变*/
05    background-image:-moz-linear-gradient(0deg,#E1DEB0 50%,transparent
06    50%,transparent);
07    background-image:-o-linear-gradient(0deg,#E1DEB0 50%,transparent 50%,transparent);
08    background-image:linear-gradient(90deg,#E1DEB0 50%,transparent 50%,transparent);
09    background-size:10% 100%;                        /*调整背景大小*/
10  }
```

实现的思路与第2章中 CSS 动画边框一致。第3~8行使用 CSS 渐变做出一个如图4.9所示的背景，然后通过第9行将背景图设置较小，使之平铺即可实现图4.8所示的效果。

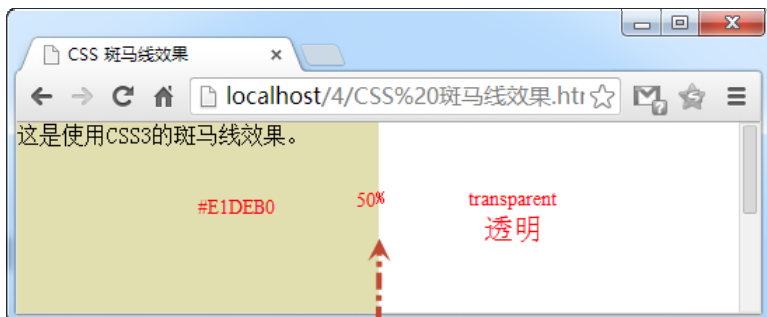


图 4.9 仅使用渐变时得到的效果

注意：第3行属性影响 Chrome 浏览器、Safari 浏览器和新版本的 Opera 浏览器（Opera 浏览器现已使用 webkit 内核），第5行影响 Firefox 浏览器，第7行兼容老版本的 Opera 浏览器，第8行影响 IE 10 及其以上版本的浏览器（IE 新版本废除了 IE 9 的 filter 滤镜，更加标准化）。当带前缀代码与不带前缀代码同时存在时，多数浏览器将以后出现的代码为准，在此处直接使用第8行。

还有一种情况是针对表格的效果，用户很难分清表格中的每一行时，添加下面的代码可以使每一行的位置更加清晰明确，这段代码使用了 CSS 伪类选择器，设置奇数行（偶数行）的 CSS 样式。

```
01 tr:nth-child(odd) {background-color: #ccc;} /*奇数行*/
02 tr:nth-child(even) {background-color: #ccc;} /*偶数行*/
```

4.6 棋盘背景

棋盘背景似乎比斑马线更具实用性，本节实例的效果图如图4.10所示。这里的棋盘背景是国际象棋的样式。

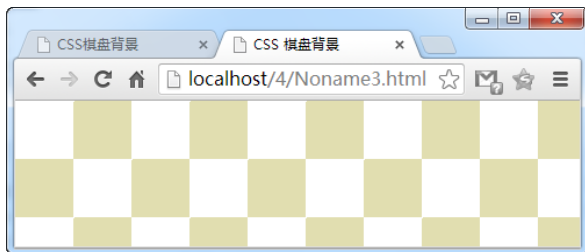


图 4.10 CSS 棋盘背景

在实现棋盘效果的过程中，最容易想到使用 CSS 斑马线背景合成——横斑马线图和竖斑马线图合成，如：

```
background-image:-webkit-linear-gradient(0deg,#E1DEB0 50%,transparent 50%,transparent),
```

```
-webkit-linear-gradient(90deg,#E1DEB0 50%,transparent 50%,transparent);
```

结果得到的效果如图 4.11 所示。这种情况不是想要的最终结果，笔者暂且称其为一种 CSS 格子背景，如果某种场景可以用的上，那么尽管这么写。至于棋盘效果，只能再寻途径，接下来想到的是第 2 章 CSS 动画边框中的斜渐变，使用两幅图叠加。首先使用下面的 CSS 代码做一个斜渐变，然后使背景图变小平铺，得到图 4.12 所示的效果。

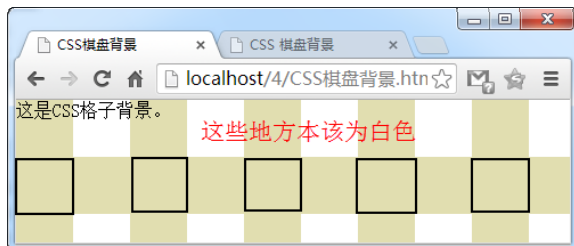


图 4.11 一种格子背景

```
01 body {
02   background-image: linear-gradient(45deg, #E1DEB0 25%, transparent 25%, transparent 03
      75%, #E1DEB0 75%, #E1DEB0),
04   linear-gradient(45deg, #E1DEB0 25%, transparent 25%, transparent 75%, #E1DEB0 75%, 05
      #E1DEB0);
06   background-size: 100px 100px;
07 }
```



图 4.12 两幅重叠的图像

现在看来根本看不出这是两幅图像，因为两幅图像是完全一致的，接下来把其中的一幅图像向右下角偏移一块距离，目的是实现图 4.13 所示的操作，标号为 1 处的三角形移动到标号为 2 的地方，使其与其他的三角形组成一个正方形。

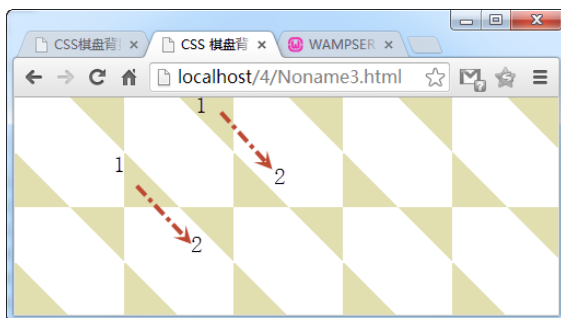


图 4.13 图像的偏移与组合过程

此过程体现在 CSS 代码上是：

```
background-position: 0 0, 50px 50px;
```

运行完此句即可实现棋盘效果。最后需要对各种浏览器做兼容。为了解决 IE 浏览器和 Firefox 浏览器下存在的严重 bug——如果背景不是黑色，中间存在的黑色斜实线严重影响美观（理论上不会出现 bug，可能是浏览器的问题），所以在背景不是黑白相间时，最终的代码建议如下：

```
01 body {
02   background-image: linear-gradient(45deg, #E1DEB0 25%, transparent 25%, transparent
03   75%, #E1DEB0 75%, #E1DEB0), /*在出现 bug 时，采用控制台寻找最合适的百分比数值*/
04   linear-gradient(45deg, #E1DEB0 26%, transparent 26%, transparent 74%, #E1DEB0
05   74%, #E1DEB0);
06   background-image: -webkit-linear-gradient(45deg, #E1DEB0 25%, transparent 25%,
07   transparent 75%, #E1DEB0 75%, #E1DEB0),
08   -webkit-linear-gradient(45deg, #E1DEB0 26%, transparent 26%, transparent 74%,
09   #E1DEB0 74%, #E1DEB0);
10   background-image: -moz-linear-gradient(45deg, #E1DEB0 24%, transparent 24%,
11   transparent 76%, #E1DEB0 76%, #E1DEB0),
12   -moz-linear-gradient(45deg, #E1DEB0 26%, transparent 26%, transparent 74%,
13   #E1DEB0 74%, #E1DEB0);
14   background-image: -o-linear-gradient(45deg, #E1DEB0 25%, transparent 25%,
15   transparent 75%, #E1DEB0 75%, #E1DEB0),
16   -o-linear-gradient(45deg, #E1DEB0 25%, transparent 25%, transparent 75%, #E1DEB0
17   75%, #E1DEB0); /*各种浏览器支持的语法都列出*/
18   background-size: 100px 100px;
19   background-position: 0 0, 50px 50px;
20 }
```

可以观察到上述的代码只是参数上的不同，在多种参数的权衡之下实验出兼容各种浏览器的最佳方案。

注意：在做兼容时，浏览器下的调试模式是一个很好的工具。

4.7 易拉罐效果

使用纯 CSS 代码就可以创建一个使用滚动条控制的易拉罐立体效果，无须任何 JavaScript 代码，仅仅使用 CSS 背景固定的属性，该效果如图 4.14 所示。

简单地说，该效果由两部分合成：一部分为固定（fixed）的背景，一部分为上方半透明的遮罩部分（易拉罐的外形）。内容部分比外层容器宽使得滚动条出现，当拖动滚动条时，背景固定，而遮罩部分会随着文档移动，这时能够显示的背景区域就改变了。该实例使用的是易拉罐外包装平面图向立体效果转换的思路。为了使立体效果更加明显，而不是生硬地改变背景区域，易拉罐的遮罩部分采用一系列左浮动的 p 标签，通过改变每个 p 标签的背景位值（background-position）制作出立体的效果。



图 4.14 纯 CSS 易拉罐效果

HTML 部分的代码如下：

```

01 <div class="coke_box">
02   <div class="coke_box_in">
03     <div id="appendHTML">
04       多个 p 标签
05     </div>
06     
07   </div>
08 </div>

```

为了更清楚地展示 p 标签的布局情况，图 4.15 可以帮助理解，该图为 firebug 下 CSS 样式的立体分析效果，图中所示的小长方体即为其中的一个 p 标签，可以看到，所有的 p 标签宽度不一致，并列在一起组成遮罩层。



图 4.15 立体样式分析

根据上述原理，得到如下 CSS 代码：

```

01 .coke_box{width:510px; height:400px; overflow:auto;}           /*外层容器*/
02 .coke_box_in{width:660px; padding-left:300px;}                /*内层文档*/
03 .coke_box_in p{width:1px; height:336px; background:url(img/coke_bg.jpg)
04   repeat-x fixed 0 0; float:left;}                             /*遮罩层 p 标签，背景固定*/
05 .coke_box_in img{margin-top:10px; border:solid white;
06   margin-left:-172px; border-right-width:300px;}               /*易拉罐外形图（半透明）*/

```



```

07 /*以下部分为遮罩层 p 标签的背景位置设置，增强易拉罐边缘的立体感，具体数据需要测算*/
08 #x1{background-position:5px 30px;}
09 #x2{background-position:0px 30px;}
10 #x3{background-position:-3px 30px;}      /*其余部分基本一致，在这里省略，见代码部分*/

```

注意：仅仅使用 CSS 的背景属性就可以做出这么炫的效果，本实例取胜点不在于代码的复杂程度，而在于设计思路。实际上，许多吸引眼球的作品都来源于好的设计，换句话说，CSS 是一种用来描述画面（网页样式）的语言，效果的好坏与设计者脑中的设计图有关。

4.8 页面顶部阴影

在网页的顶部设置一个阴影区域可以使得网页效果不单调。对 body 使用 before 伪类，向顶部增加阴影，增强美观度，如图 4.16 所示。

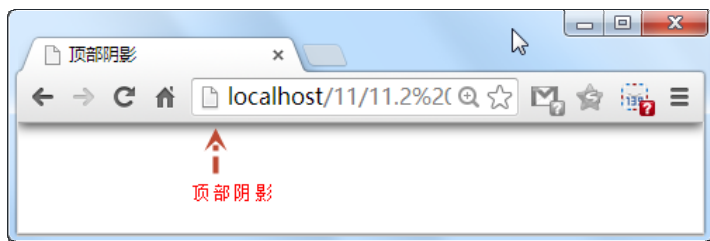


图 4.16 页面顶部阴影

CSS 代码如下：

```

01 body:before {
02   content: "";
03   position: fixed;top: -10px;left: 0;
04   width: 100%;height: 10px;
05   box-shadow: 0px 0px 10px rgba(0,0,0,.8);      /*使用阴影属性即可*/
06   z-index: 100;
07 }

```



第 5 章 变换与动画

随着多个浏览器的不断发展和进步，除去 IE6~8 和使用它们作为内核的浏览器之外，多数的浏览器日益走向标准化之路，更加符合 W3C 标准。在这种情况下，网页的设计由单纯的静态网页，到使用 JavaScript 结合制作具有动态效果的网页，再到现如今使用纯 CSS 3 代码实现动画效果的网页，并且 CSS 3 逐渐流行起来。对于网页来说，适当地使用变换和动画使网页拜托静态无疑是好的，伴随新的 CSS 3 支持和 JavaScript 的增强，越来越多的网页向动画效果丰富的网页改进。

在多种技术的支持下，可以使用多样的方法实现变换与动画的效果。本章将在 JavaScript 与 CSS 结合的方式和纯 CSS 方式实现动画方面列举多个变换与动画方面的实例，在图片一章中关于图片轮播图的动画效果不在本章重叠。

本章主要涉及的内容有：

- CSS 3 变换与动画的基础及实例
- 纸张边角动画、气泡式提示
- 对联广告
- 页面 loading 效果、进度条
- 图标滑动切换特效
- 流星划过与雪花飘落
- 数字滚动器、模拟时钟
- 苹果系统的 DOCK 栏和 Stack 特效
- 扇形展开
- 回到页面的顶部
- 拖曳和抛出

5.1 CSS 3 变换与动画的基础及实例

5.1.1 CSS 3 变形概述

CSS 变换是 CSS 3 中新增的属性，拥有 2D 变换和 3D 变换两个方面。变换包括变形和转换两个部分，变形的关键字为 **transform**，转化的关键词为 **transition**。在之前的拍立得效果框中已经用到过旋转变形，作为很多实例尤其是动画相关的实例的基础，有必要对 CSS 3

中的新属性做一下详解。在 CSS 3 变形这一节，重点对 CSS 3 transform 中对 2D 变形的属性进行应用。

目前主流浏览器对变换已有了很好的支持，IE 浏览器、Opera 浏览器、Chrome 浏览器、Safari 浏览器和 Firefox 浏览器均支持 transform，只是在 IE9 上需要加-ms-前缀，在旧版本的 Firefox 浏览器上需要加-moz-前缀，在 Opera 浏览器上需要加-o-前缀，在 Chrome 浏览器和 Safari 浏览器上需要加-webkit-前缀。在最新版本的标准浏览器上不需要加前缀。

5.1.2 CSS 3 变形语法详解及应用

在 2D 变形时首先要指定一个变形的基点，变形的过程将以该基点为基础进行改变，同样的代码在变形基点不同时得到的变换结果是不相同的。比如一个正方形绕它的左上角旋转 30 度和绕它的右下角旋转 30 度得到的效果是不相同的。实际上这个变换基点是变换坐标系统的原点位置。在 CSS 中，制定变换基点的属性为 transform-origin（注意前缀的使用，有些浏览器的 transform-origin 不适用于 3D 转换），在不设置该属性时，将以元素的中心（50% 50% 0）为中心做变换。其语法为：

```
transform-origin: x 轴参数 y 轴参数 ?z 轴参数;
```

2D 变换中不需要 z 轴参数。参数可选择 3 类：一类为关键词 left、center、right（x 轴）、top、center、bottom（y 轴）；一类为百分比，可以使用负数；还有一类为长度值，可以为负数。z 轴的参数只可以使用长度值。

在变形实例中将对边长为 100 像素的正方形使用默认值（元素中心）进行变形。变形包含平移（translate）、旋转（rotate）、缩放（scale）、错切（skew）以及多种变换类型的组合（matrix）。

- 平移（translate）

```
transform: translate(x,y);
```

x、y 的值为长度值，如 x=50px，y=50px，图形将从左侧移动 50 像素、从顶部移动 50 像素，变换的结果如图 5.1 所示。可以使用负数，对应有 translateX(x)、translateY(y)方法。

- 旋转（rotate）

```
transform: rotate(deg);
```

deg 为旋转度数，如 deg=45deg，图形将绕基点位置顺时针旋转 45 度，如图 5.2 所示。可以使用负数，负数时逆时针旋转。

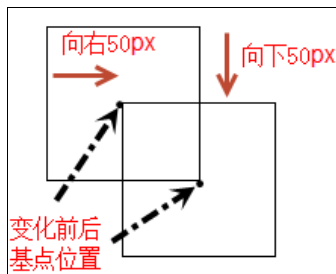


图 5.1 translate

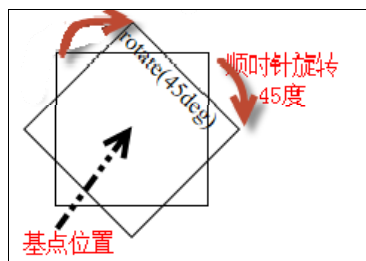


图 5.2 rotate

- 缩放 (scale)

```
transform:scale(x,y);
```

x,y 为数值, y 省略时 y=x。变形的效果是在 x 轴方向上缩放 x 倍, 在 y 轴方向上缩放 y 倍, 如 x=1.5、y=1.2, 效果如图 5.3 所示。可以使用负值, 对应有 scaleX(x)、scaleY(y) 方法。

- 错切 (skew)

```
transform:skew(x,y);
```

错切又称扭曲、翻转, x 与 y 取角度值。得到元素在 x 轴方向 (从底部向底部看逆时针) 旋转 x 度、在 y 轴方向 (从左向右看逆时针) 旋转 y 度后的平面效果。如 x=30deg、y=10deg 的效果如图 5.4 所示。可以使用负数, 对应有 skewX(x)、skewY(y) 方法。

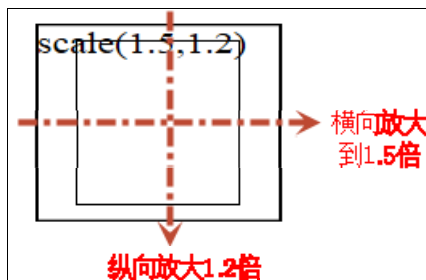


图 5.3 scale

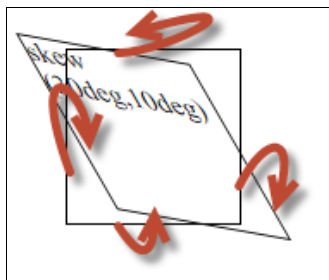


图 5.4 skew

- 混合变形 (matrix)

```
matrix(a,b,c,d,e,f);
```

对一个元素同时应用多个变形, 可以使用上述 4 种语法的结合, 如果使用一句实现多种变换, 那么只有 matrix 方法了。matrix 方法需要 6 个参数, 比较复杂。事实上, 网页对 CSS 变换的解释是通过矩阵的运算 (矩阵乘法) 来实现的, 2D 变换是多个 3 乘 3 矩阵的运算, 3D 变换是多个 4 乘 4 矩阵的运算。在配合 JavaScript 使用时, 语言的组合实现起来有很多问题, 最经常遇到的是使用 JavaScript 改变部分属性值, 覆盖所有的原始数值的问题, 这些情况下使用语法组合就不如 matrix 方法快捷了, 不过 matrix 方法也有数值计算复杂、数值难以控制 (有小数) 等问题。由于复杂性, 这里不说明 matrix 方法的使用, 若要了解可以参考其他资料 (参考资料: <http://www.renjianzhi.cn/16.html>)。

注意: 3D 转换与 2D 转换大多思想是一致的, 本质都是矩阵变换, 2D 的理解可以扩展到 3D。

特别注意在浏览器中获取样式时, 在变形样式的返回值为 matrix(a,b,c,d,e,f), 并不是返回单一的属性值, 但多数情况下, 利用 CSS 转换和 JavaScript 设置, 并不影响实际使用。

5.1.3 CSS 3 转换概述

所谓 CSS 3 转换, 是在不使用 Flash 动画或 JavaScript 的情况下, 当元素从一种样式变换为另一种样式时为元素添加效果, 该属性是 CSS 变换的一部分。CSS 3 转换是元素在 CSS

样式改变时获得平滑切换效果，也就是说让网页元素的 CSS 值变化变得平滑，看起来像动画一般。在 CSS3 中这一属性的关键词为 transition。

主流浏览器对转换属性已有很好的支持。IE 10、Firefox、Chrome 以及 Opera 支持 transition 属性。Safari 需要前缀 -webkit-，Chrome 25 以及更早的版本，需要前缀 -webkit-，IE 9 以及更早的版本，不支持 transition 属性。

在没有 CSS 3 转换之前，涉及样式平滑变换的方法为 JavaScript 制作的元素运动，如本书中经常使用的运动框架，通过 JavaScript 中设置定时器，一点点地实现 CSS 属性值的改变，而应用 CSS 3 新属性，一切都变得很简单，不用担心 JavaScript 在数字处理上的复杂和不安全，并且可以使用数值控制样式之间的切换的时间，没有理由不喜欢它。

5.1.4 CSS 3 转换语法详解

表 5.1 列出了 CSS 转换的所有属性及其参数情况。

表 5.1 转换属性

属性	属性描述
transition-property	要应用转换的CSS属性的名称
transition-duration	转换时间，单位：s或ms
transition-timing-function	转换效果的时间曲线，默认值ease
transition-delay	有没有延迟，默认值0，单位：s或ms
transition	用于以上4个属性的简写，即以上4个属性值的组合

- transition-property

transition-property: none|all|property;

当参数值为 none 时，没有任何样式会得到转换效果，当参数值为 all 时，元素的所有样式都会得到转换效果，当参数值为 CSS 样式（如 height、width 等）列表时，这些自定义的 CSS 属性将得到转换效果。多个属性用 “,” 隔开。

- transition-duration

transition-duration: time;

time 为转换时间，如 2s。

- transition-timing-function

transition-timing-function: linear|ease|ease-in|ease-out|ease-in-out|cubic-bezier(n,n,n,n);

转换效果曲线影响着样式转换的动画效果，下表为参数的取值和效果列表，CSS 3 转换与 CSS 3 动画在很多地方是相似的。

表 5.2 CSS 3 转换效果曲线

取值	效果	与哪种贝济埃曲线等价
linear	匀速	cubic-bezier(0,0,1,1)
ease	慢速开始，然后变快，然后慢速结束	cubic-bezier(0.25,0.1,0.25,1)
ease-in	慢速开始，后匀速结束	cubic-bezier(0.42,0,1,1)
ease-out	匀速开始，后慢速结束	cubic-bezier(0,0,0.58,1)
ease-in-out	慢速开始，中间匀速，后慢速结束	cubic-bezier(0.42,0,0.58,1)
cubic-bezier(n,n,n,n)	n取0~1之间的值，贝济埃函数的效果	本身

注意：ease 效果并不等同于 ease-in-out 效果，虽然肉眼分辨困难，但根据 cubic-bezier 函数的取值可知两者不同。

- transition-delay

```
transition-delay: time;
```

time 为转换开始前的等待时间，如 2s。

- transition 简写属性

```
transition: property duration timing-function delay;
```

把以上的属性按顺序结合起来即可，后两个参数可以按照默认设置，默认设置时使用的是默认值。但 transition-duration 属性必须设置，否则将不会得到任何效果。

5.1.5 CSS 3 转换具体实例

转换属性针对的是不同样式的转换，与动画不同，转换需要事件触发（如: hover），在想要应用转换的元素的样式中用上述语法声明该元素上应用转换，并指明哪些 CSS 样式获得转换效果。在触发时重新定义 CSS 样式属性即可让已经声明过使用转换效果的元素从初始的 CSS 样式平滑地转换到触发时的样式。如以下代码：

```
01 div{
02   height:200px;
03   width:200px;
04   background:#3994C9;
05   transition:width 2s ease;
06 }
07 div:hover{
08   width:400px;
09 }
```

在第 5 行声明 div 在宽度上应用转换效果，在第 8 行指定鼠标悬停在 div 上方时的宽度为 400 像素，则 div 的宽度将由初始的 200 像素（第 3 行）缓慢地变化到 400 像素。

使用 CSS 3 转换可以不涉及 JavaScript 并且比 JavaScript 实现更为简单，还可以解决当前 JavaScript 和 jQuery 等对 CSS 3 中属性的不完善的支持问题，可见 CSS 3 在无论在动画方面、独立性方面和性能方面都做了很大的改进。

5.1.6 CSS 3 动画概述

CSS 3 动画指的是使用 CSS 代码，使得网页中的元素运动起来形成动画。这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript。从某种程度上讲，CSS 3 过渡效果也是动画，只不过这种动画只动了 1 次，而用 CSS 3 动画属性实现的效果，是多次不同的轨迹的组合，并且可以设置播放次数，还可以控制播放与暂停。

CSS 3 动画有两个关键词，一个是 @keyframes，它的作用是创建一系列的动画；另一个是 animation，它的作用是规定 HTML 中哪个元素使用哪种动画，即应用动画。

版本较高的浏览器对 CSS 3 动画已经有了很好的支持，特别说明的是，改用 webkit 内核的 Opera 浏览器在 CSS 3 动画方面表现优异。下面是主流浏览器对 CSS 3 动画属性的支持情况。

- IE 10、Firefox、Chrome 以及 Opera 支持@keyframes 规则和 animation 属性。
- Firefox 支持替代的@-moz-keyframes 规则。
- Opera 支持替代的@-o-keyframes 规则。
- Safari 和 Chrome 支持替代的@-webkit-keyframes 和-webkit-animation 属性。
- IE 9 以及更早的版本，不支持@keyframes 规则和 animation 属性。

注意：为了取得很好的兼容性，建议带前缀的代码全部列在前面，无前缀的代码放在后面。

5.1.7 CSS 3 动画语法详解

CSS 3 动画属性见表 5.3。

表 5.3 CSS 3 动画属性表

属性名称	属性作用
@keyframes	定义一个动画
animation-name	要应用到元素上的动画的名称
animation-duration	动画完成一个过程的时间
animation-timing-function	动画速度曲线，影响运动的类型。默认是ease
animation-delay	动画开始的时间，即是否设置延迟
animation-iteration-count	动画被播放的次数。默认是 1
animation-direction	规定动画是否在下一周期逆向地播放。默认是 “normal”
animation-play-state	规定动画的运行或暂停。默认是running
animation-fill-mode	规定对象动画时间之外的状态
animation 简写属性	参数表列为上面除去@keyframes、animation-play-state、animation-fill-mode之外的6个属性的值

上表为 CSS 3 动画里所有属性的列表，下面将一一说明表中属性的用法。

- @keyframes

```
@keyframes animationname {keyframes-selector {css-styles;}}
```

即

```
@keyframes 定义动画的名称 {阶段 1{CSS 样式表列 1}[【 可选 】阶段 2{CSS 样式表列 2} ... ]}
```

每个阶段用百分比来表示，从 0%~100%，0%和 100%必须设置，如果仅有开始和结束两个阶段，可以写为 from 和 to，他们分别与 0%和 100%等价。例如：

```
01 @keyframes mymove
02 {
03   0% {top:0px;left:0px;transform:rotate(50deg);}
04   10% {top:100px;left:50px;transform:rotate(80deg);}
05   20% {top:300px;left:40px;transform:rotate(-50deg);}
06   30% {top:400px;left:200px;transform:rotate(150deg);}
```

```
07 40% {top:700px;left:300px;transform:rotate(350deg);}
08 100% {top:200px;left:50px;transform:rotate(160deg);}
09 }
```

或者:

```
01 @keyframes mymove
02 {
03   from {top:0px;left:0px;transform:rotate(50deg);}
04   to {top:100px;left:50px;transform:rotate(80deg);}
05 }
```

- animation-name

```
animation-name: keyframename|none;
```

当值为 none 时，没有动画效果（可以用于覆盖来自于级联的动画），其他情况下，直接使用@keyframes 定义过的动画的名称。

- animation-duration

```
animation-duration: time;
```

time 为执行一次动画所需的时间，单位为 s 或 ms。

- animation-timing-function

```
transition-timing-function: linear|ease|ease-in|ease-out|ease-in-out|cubic-bezier(n,n,n,n);
```

运动曲线，参数取值及意义见表 5.4。

表 5.4 animation-timing-function取值

取值	效果	与哪种贝济埃曲线等价
linear	匀速	cubic-bezier(0,0,1,1)
ease	慢速开始，然后变快，然后慢速结束	cubic-bezier(0.25,0.1,0.25,1)
ease-in	慢速开始，后匀速结束	cubic-bezier(0.42,0,1,1)
ease-out	匀速开始，后慢速结束	cubic-bezier(0,0,0.58,1)
ease-in-out	慢速开始，中间匀速，后慢速结束	cubic-bezier(0.42,0,0.58,1)
cubic-bezier(n,n,n,n)	n取0~1之间的值，贝济埃函数的效果	本身

- animation-delay

```
animation-delay: time;
```

time 为开始动画之前的等待时间，也就是延迟，单位是 s 或 ms。

- animation-iteration-count

```
animation-iteration-count: n|infinite;
```

动画播放次数，当值为正整数时，播放次数为该数值。当值为 infinite 时，无限播放。

- animation-direction

```
animation-direction: normal|alternate;
```

当值为 normal 时，正常播放，也就是一次动画结束时回到动画开始的位置继续播放。当值为 alternate 时，逆向循环播放，也就是说一次动画结束时，从结束位置做逆向运动回到动画的开始位置，然后继续运动。

- animation-play-state

```
animation-play-state: paused|running;
```


当值为 `paused` 时，动画暂停，停在最后一刻的位置。当值为 `running` 时，从暂停的位置继续运动。

- `animation-fill-mode`

`animation-fill-mode : none | forwards | backwards | both;`

`animation-fill-mode` 属性规定动画在播放之前或之后，其动画效果是否可见。当值为 `none` 时，不改变默认的行为，动画完成之后，元素会回到最初的（即动画之前的）CSS 样式。当值为 `forwards`，在动画完成之后，保持最后的 CSS 样式不变。当值为 `backwards` 时，在动画完成之后回到最初的（即动画之前的）CSS 样式。当值为 `both` 时，向前和向后填充模式都被应用。

- `animation`

`animation` 属性是一个简写属性，用于按顺序依次构造 `animation-name`、`animation-duration`、`animation-timing-function`、`animation-delay`、`animation-iteration-count`、`animation-direction` 这 6 个属性。

注意：前两个属性一定要设置，否则不会出现效果。

5.1.8 简单实例

```
01 div
02 {
03   -moz-animation: myfirst 5s linear 2s infinite alternate;      /* Firefox: */
04   -webkit-animation: myfirst 5s linear 2s infinite alternate;   /* Safari 和 Chrome: */
05   -o-animation: myfirst 5s linear 2s infinite alternate;        /* Opera: */
06   animation: myfirst 5s linear 2s infinite alternate;
07 }
```

`div` 将在延迟 2s 之后以 5s 为一个周期、按照匀速的方式无限次地执行名称为 `myfirst` 的动画，并且连续的两次分别为正向播放和逆向播放。因动画效果无法使用图片说明，具体效果请打开实例查看。

5.2 纸张边角动画效果

在网页设计中，特别突出纸张效果的网页多见于文字阅读类的新型网站，同时在某些网站中采取纸张的卷曲和边角翻起来加强视觉体验，常见的是 QQ 空间右上角的广告部分、新浪博客个人主页右上角的换肤功能以及维基百科上的纸张效果。静态的纸张页面通常通过 CSS 3 圆角与 CSS 3 渐变来实现，较为简单，本节的实例有两种：一种是纸张边角稍稍卷起，另一种是边角翻折效果。目的是模仿一下上面提到的名站效果。

5.2.1 纸张边角稍稍卷起

使用 CSS 3 border-top-right-radius 属性和渐变效果, IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-top-right-radius 属性。IE 低版本无法直接使用 CSS 3 来实现。本实例的效果如图 5.5 所示。



图 5.5 纸张微卷效果

HTML 部分定义一个 paper 类 div 内含一个 content 类的 div 供以后使用。content 类的 div 里随意写网页内容。clear 类的 div 用来清除浮动。

```
01 <div class="paper">
02   <div class="content">
03       <h1>《背影》——朱自清</h1>
04       <p>
05           我与父亲不相见已二年余了，我最不能忘记的是他的背影。
06       </p>
07   </div>
08   <div class="clear"></div>
09 </div>
```

使用下面的 CSS 代码：

```
01 *{
02     margin:0;
03     padding:0;
04 }
05 .clear{
06     clear:both;                /*防止浮动带来的混乱*/
07 }
08 .paper{
09     position:relative;
10     height:auto;
11     width:860px;
12     margin:30px auto;
13     padding:15px;
14     border:1px solid #CCC;      /*阴影效果*/
15     box-shadow:2px 1px 4px 2px rgba(0,0,0,0.3);
16     -webkit-transition:all 0.4s; /* Safari 和 Chrome */
17     -moz-transition:all 0.4s;   /* 火狐 */
18     -ms-transition:all 0.4s;    /* IE */
19     -o-transition:all 0.4s;     /*其他支持他的浏览器 */
```

```

20 }
21 .paper:hover{
22   border-top-right-radius:140px 50px; /*纸张卷边实质是圆角的效果*/
23 }
24 .content{
25     float:left;
26     width:100%;
27     height:auto;
28     min-height:760px;                /*设置纸张的最小高度，不是以文字的多少适配*/
29 }

```

第16行到第19行使不同浏览器产生动画渐变效果，这种效果与jQuery中animate作用相同，当鼠标悬浮事件被触发时，CSS属性在0.4s内实现样式的变化，最终样式为hover伪类中的属性内容。如果没有这些，纸张卷边非常突然。border-top-right-radius向纸张元素的右上角添加圆角边框，加上设置的外阴影，产生一种立体感。这段代码执行之后的效果如图5.6所示，因为纸张仅有一张，立体感不是太强，为了增强整体的立体感，最好使用下面的CSS代码在paper类div的前后添加两张页面。



图 5.6 一张纸立体感不好

```

01 .paper:before, .paper:after{
02   content: "";
03   background:-moz-linear-gradient(top, #ffffff 0%, #e5e5e5 100%); /*线性渐变*/
04   background:-webkit-gradient(linear, left top, left bottom, color-stop(0%,#ffffff),
05     color-stop(100%,#e5e5e5)); /*webkit 老语法*/
06   background:-webkit-linear-gradient(top, #ffffff 0%,#e5e5e5 100%); /*webkit*/
07   background:-o-linear-gradient(top, #ffffff 0%,#e5e5e5 100%); /* Opera 11.10+ */
08   background:-ms-linear-gradient(top, #ffffff 0%,#e5e5e5 100%); /* IE10+ */
09   background:linear-gradient(top, #ffffff 0%,#e5e5e5 100%); /* W3C */
10   background:#FFF;
11   border:1px solid #CCC;
12   position: absolute;
13   z-index: -1; /*显示优先级*/
14   box-shadow:2px 1px 4px 2px rgba(0,0,0,0.3);
15 }
16 .paper:before {
17   height:96%;
18   width:96%; /*底部纸张的缩放*/
19   top:-20px;
20   left:20px;

```

```

21 }
22 .paper:after {
23     height:96%;
24     width:98%;
25     top:-10px;
26     left:10px;
27 }

```

前后添加的两张纸张是通过 before 和 after 伪类实现的。在上述代码的第 1~15 行，渐变效果设置的是纸面，与页面数量无关。

5.2.2 边角翻折

鉴于边角微微卷起在立体感方面表现不出色，使用边角翻折可以解决此问题。边角翻折的效果图如图 5.7 所示，这一实例可以应用在右上角换肤功能和右上角小广告上。



图 5.7 纸张翻折

本例的实现是通过右上角附加一块 div 实现。HTML、CSS 部分的代码如下：

```

//HTML
01 <div id="paper_top_right">
02 <div id="paper_top_right_animate"></div>
03 <!--广告图、连接、文本的放置区-->
04 </div>
//CSS
01 *{margin:0;padding:0;}
02 body{
03     background:#D9D9D9;
04 }
05 #paper_top_right{                                /*整体*/
06     position:absolute;
07     height:150px;
08     width:150px;
09     top:0;                                        /*右上角*/
10     right:0;
11 }
12 #paper_top_right_animate{

```

```

13 height:30px;
14 width:30px;
15 position:absolute;
16 background:linear-gradient(45deg,#D7D7D7,#E2E2E2 20%,#FFFFFF 35%,#D5D5D5
17 50%,#555555 50%,#555555);
18 background:-webkit-linear-gradient(45deg,#D7D7D7,#E2E2E2 20%,#FFFFFF
19 35%,#D5D5D5 50%,#555555 50%,#555555 );
20 background:-o-linear-gradient(45deg,#D7D7D7,#E2E2E2 20%,#FFFFFF 35%,#D5D5D5
21 50%,#555555 50%,#555555);
22 background:-moz-linear-gradient(45deg,#D7D7D7,#E2E2E2 20%,#FFFFFF
23 35%,#D5D5D5 50%,#555555 50%,#555555 ); /*利用渐变做立体效果*/
24 border-bottom-left-radius:10px 90px;
25 border-bottom-right-radius:100px 1px;
26 box-shadow:-3px 2px 5px #333333; /*阴影*/
27 top:0; /*相对定位*/
28 right:0;
29 z-index:1;
30 }

```

边角部分的效果是由第 16~23 行实现的,使用 CSS 的渐变属性,沿 45 度渐近线方向,前半部分为边角,后半部分的黑色区域看起来像是纸张下面的背景。在边角部分,35%处的白色高亮部分是为增强立体感而设置的。在 24、25 行,通过设置部分圆角,使得边角有卷曲感。第 26 行设置外阴影增强立体感,因为边角部分是在右上角,所以 box-shadow 的水平偏移为负值、垂直偏移为正直才符合要求。在定位时,div 的右上角固定在 body 区域的右上角,因为在做渐变时使用的是百分比,所以通过调整 paper_top_right_animate 的大小即可改变边角翻折的程度,如果把大小属性与 JavaScript 结合起来,即可实现边角翻折的动画。下面的这段 jQuery 代码即可实现此功能。

```

01 $(function (){
02     $("#paper_top_right").hover(function(){show_top_right();},function(){hide_top_right();});
03 });
04 function show_top_right(){ //animate 展示
05     $("#paper_top_right_animate").stop(true,false).animate
06     ({height:"150px",width:"150px"},"slow");
07 }
08 function hide_top_right(){ //animate 隐藏
09     $("#paper_top_right_animate").stop(true,false).animate
10     ({height:"30px",width:"30px"},"slow");
11 }

```

注意: 在此段 jQuery 代码中请注意 stop 函数的使用,stop 函数可以解决鼠标频繁移动时的错误。

5.2.3 更具立体感的边角翻折效果

如果以上的两种方案都不满足立体感的要求，则可以选择这种方案。此方案的效果图如图 5.8 所示。

CSS 只提供了圆角的实现方式，但却没有提供曲线的实现方式，对于图 5.8 中的曲线，可以通过分块的组合来实现，图 5.9 是这种组合的原型图。图中带有标号部分的圆弧是通过部分圆角得来的。由于这张方案是组合，所以需要的 div 较多，HTML 部分为下面的代码。



图 5.8 纸张边角翻折

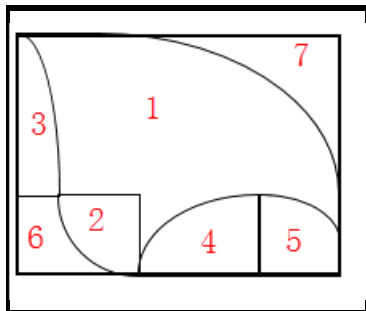


图 5.9 原型图

```
01 <div id="top_right">
02 <div id="top_right_all">
03 <div id="top_left"></div>
04 <div id="bottom_left"><div id="bottom_left_right"></div></div>
05 <div id="bottom_middle"></div>
06 <div id="bottom_right"></div>
07 </div>
08 </div>
```

在 CSS 部分，需要做的关键设置是使用绝对定位确定每个小块的位置。在颜色的选择上，需要对 1、2 两处设置渐变色，3~6 处设置纸张的背景色，7 处设置为纸张下面的背景色。颜色选择好以后，为了增强立体感，还需要设置边框阴影，有些 div 的边框阴影为外阴影，有些这是内阴影，各部分阴影的参数也有不同，最终的目的是让他们组成一个整体。在做完以上工作之后，把实线边框隐藏即可。所有工作的合集体现在 CSS 代码上是下面这样的。

```
01 body{
02   background:#D9D9D9;
03 }
04 #top_right{                               /*整体*/
05   width:200px;
06   height:150px;
07   background:#555555;
08   position:absolute;
09   top:0;
```

```

10  right:0;
11  }
12  #top_right_all{                /*1 位置*/
13      width:200px;
14      height:150px;
15      border-top-right-radius:200px 150px;
16      position:absolute;
17      background:linear-gradient(45deg,#D7D7D7,#E2E2E2 20%,#FFFFFF 50%,#E2E2E2
18  75%,#E2E2E2);
19      background:-webkit-linear-gradient(45deg,#D7D7D7,#E2E2E2 20%,#FFFFFF
20  50%,#E2E2E2 75%,#E2E2E2);
21  }
22  #top_left{                    /*3 位置*/
23      width:25px;
24      height:100px;
25      position:absolute;
26      top:0;
27      left:0;
28      border-top-right-radius:25px 100px;
29      background:#D9D9D9;        /*注意与背景色相同*/
30      box-shadow:-3px 2px 2px #333333 inset;
31  }
32  #bottom_left{                /*2、6 位置*/
33      width:75px;
34      height:50px;
35      position:absolute;
36      top:100px;
37      left:0;
38      background:#FFFFFF;
39      border:0;
40      background:#D9D9D9;        /*注意与背景色相同*/
41  }
42  #bottom_left_right{          /*2 位置*/
43      width:50px;
44      height:50px;
45      position:absolute;
46      top:0;
47      left:25px;
48      border-bottom-left-radius:50px 50px;
49      background:linear-gradient(45deg,#E8E8E8,#F0F0F0);
50      background:-webkit-linear-gradient(45deg,#E8E8E8,#F0F0F0);
51      border-top-width:0;
52      box-shadow:-3px 3px 2px #333333 ;

```

```

53 }
54 #bottom_middle{                      /*4 位置*/
55     width:75px;
56     height:50px;
57     position:absolute;
58     top:100px;
59     left:75px;
60     border-top-left-radius:75px 50px;
61     background:#FFFFFF;
62     box-shadow:2px 3px 2px #333333 inset;
63     background:#D9D9D9;                /*注意与背景色相同*/
64 }
65 #bottom_right{                        /*5 位置*/
66     width:50px;
67     height:50px;
68     position:absolute;
69     top:100px;
70     left:150px;
71     border-top-right-radius:50px 50px;
72     background:#FFFFFF;
73     box-shadow:-5px 3px 2px #333333 inset;
74     background:#D9D9D9;                /*注意与背景色相同*/
75 }

```

注意：

(1) 在做定位时，将边框全部展示便于理解和进行定位布局，如果在定位方面不太熟悉或者定位较为复杂，建议使边框可见以帮助定位。

(2) 在本例中，为了使读者便于理解各部分的拼接以及更简单的取值，实例中使用的是具体的数值，在移植向其他场景时需要修改的参数较多，如果需要加强代码重用性，建议 CSS 部分不直接写，用 JavaScript 代码动态计算出合理的取值之后，使用 JavaScript 将 CSS 部分写入网页。

(3) 本方案经过各种主流浏览器的兼容之后，仍存在细节瑕疵之处，原因在于小块的数量太多。在很多场景下，尽可能地减少元素的数量是很有益的。

(4) 整体分块然后组合还可应用于多种场景，例如纯 CSS 创建 banner、图标、画出腾讯 QQ 企鹅或者天猫的 logo 等。

5.3 气泡式提示

很多网页使用气泡提示用户更详细的内容，最新的 QQ 对话框也使用了气泡效果，气泡式提示的同义词有冒泡提示、对话气泡等。冒泡提示多使用 JavaScript 实现，但使用纯 CSS 代码也可以实现。本节只用 CSS 代码，做出如图 5.10 所示的效果。

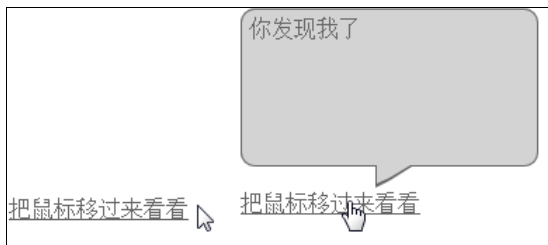


图 5.10 气泡式提示

由于纯 CSS 气泡式提示原理较为简单,使用 CSS 中: hover 把原本不显示的气泡展示出来即可,所以这里把本属于边框范畴的气泡实现融合到了本节中。在边框与图片一章中,曾经使用过 border 属性制作各类三角形,图中气泡下方的三角形就是用 border 制作的两个大小不一致的三角形重叠形成的。为了使用 hover 伪类,需将气泡 div 作为 a 标签的子元素。

```
<a href="#">把鼠标移过来看看<div id="tips">你发现我了</div></a>
```

在 CSS 部分,使用: before 和: after 伪对象属性在气泡框的上方和下方分别添加两个三角形,做好三角形的定位工作,使得所有部分的组合恰到好处。气泡默认不显示,只有当 hover 伪类选择器匹配到 a 标签才把 div 显示出来。

```
01 #tips{
02   width:200px;height:100px;padding:5px;
03   border:1px solid;border-radius:10px;
04   position:absolute;display:none;top:-130px;
05   background:#D3D3D3;
06 }
07 #tips:before{content:"";width:0;height:0;
08   border-top:10px solid ;border-bottom:5px solid transparent ;
09   border-left:10px solid ;border-right:16px solid transparent;
10   position:absolute;left:45%;top:101%;}
11 #tips:after{content:"";width:0;height:0;
12   border-top:9px solid #D3D3D3 ;border-bottom:4px solid transparent ;
13   border-left:9px solid #D3D3D3;border-right:15px solid transparent;
14   position:absolute;left:45.5%;top:100%;}
15 a{position:absolute;top:140px;}
16 a:hover>#tips{display:block;}
```

在第 7 行和第 11 行中, :before 与: after 需与 content 配合使用。否则伪对象不可见。

类似的方法,可以制作思考式气泡,如图 5.11 所示,其实现与气泡式提示的实现方式相同。

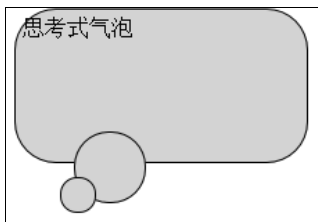


图 5.11 思考式气泡

气泡式提示的代码如下：

```

01 #tips2{
02     width:200px;height:100px;padding:5px;
03     border:1px solid;border-radius:30px;
04     position:absolute;display:block;left:400px;
05     background:#D3D3D3;
06 }
07 #tips2:before{content:"";width:50px;height:50px;background:#D3D3D3;
08     border:1px solid;border-radius:25px;
09     position:absolute;left:20%;top:80%;}
10 #tips2:after{content:"";width:24px;height:24px;background:#D3D3D3;
11     border:1px solid;border-radius:12px;
12     position:absolute;left:15%;top:110%;}

```

5.4 对联广告

很多网站收入的主要来源都在于广告上，本节实现一种最常见的对联广告的形式，实例的效果与百度推广的对联广告效果相似，打开页面后，广告会自动到达窗口的中部位置，左右各一个，页面滚动结束时广告依然会保持在窗口的中部位置，如图 5.12 所示。当单击广告上的关闭按钮时，广告不再显示。当浏览器的窗口大小调整后，广告会重新调整到合适的位置。

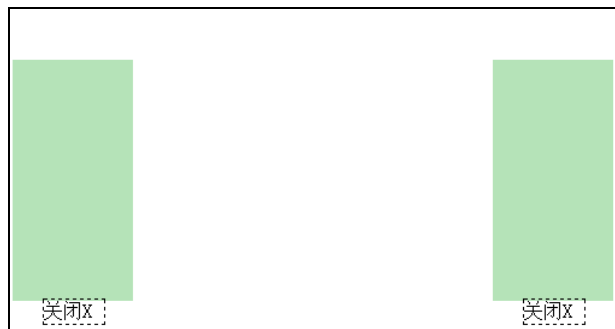


图 5.12 对联广告

在 HTML 中定义两个 div 作为广告容器，其中放置一个 span 作为关闭按钮，采用绝对定位将 span 调整到 div 的下部，同时将两个 div 分别放置在左边和右边，在 JavaScript 部分动态地计算 div 的 top 值即可。总体上讲该实例代码主要是 JavaScript：

```

01 <script type="text/javascript" src="js/zQuery.js"></script>
02 <script type="text/javascript">
03     window.onload=function (){
04         var ad1=document.getElementById('ad1');           //获取到广告一
05         var ad2=document.getElementById('ad2');           //获取到广告二
06         var span=document.getElementsByTagName('span');    //获取到关闭按钮

```

```

07 for (var i=0;i<span.length ;i++ )
08 {
09   span[i].onclick=function(){this.parentNode.style.display='none';}
10                                     //点击关闭按钮，找到该 span 的父元素 div，将其隐藏
11 }
12 var ad_height=ad1.offsetHeight;           //获取到广告宽度
13 var view_height=win('height');           //获取到窗口大小
14 var top_first=parseInt((view_height-ad_height)/2); //计算 top
15 move(ad1,{top:[top_first]});
16 move(ad2,{top:[top_first]});           //移动 div 到中间
17 window.onscroll=function (){           //页面滚动
18   var offtop=win('scrollTop');           //获取滚动的距离
19   var top=top_first+offtop;           //计算 top
20   move(ad1,{top:[top]});
21   move(ad2,{top:[top]});
22 }
23 window.onresize=function (){           //窗口调整大小
24   var offtop=win('scrollTop');
25   view_height=win('height');
26   top_first=parseInt((view_height-ad_height)/2);
27   var top=top_first+offtop;
28   move(ad1,{top:[top]});
29   move(ad2,{top:[top]});}}
30 </script>

```

注意：可扩展到客服窗口、漂浮元素、固定页脚等地。

5.5 页面 loading 效果

由于国内的网速问题，在图片密集处或者页面较大时，常常需要设置一个正在加载的图标，如图 5.13 所示，用户会在图标旋转的时候等待内容加载完成。在 CSS 3 之前，这一功能一般由一张 gif 动态图片来完成，在 CSS 3 定义了旋转属性之后，最直接的实现旋转的方式是使用一张静态图片使其旋转实现。事实上，旋转圈可以直接由 CSS 代码生成。

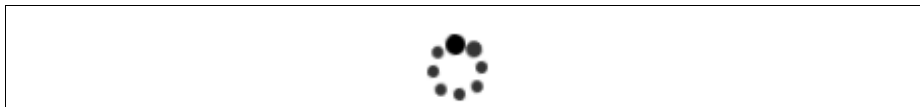


图 5.13 页面 loading 效果

在边框与图片关于边框新属性的一节中曾经讲到，box-shadow 可以用来设置阴影，并且可以设置多个阴影。图中小圆点的内部有一个小 div，它设置了圆角，是一个圆形。每个小黑点都是该 div 的多个投影，而黑点大小又可以通过阴影的参数来改变。最后配合 CSS 3

动画，可以使这个 div 无限地旋转。CSS 代码为：

```

01 div{
02   margin:100px auto;
03   width:4px;height:4px;border-radius:2px;
04   box-shadow:0 -12px 0 3px black ,           /*圆点生成，上*/
05               0 12px 0 1px #333,           /*下*/
06               -12px 0 0 1px #333,         /*左*/
07               12px 0 0 1px #333,         /*右*/
08               -9px -9px 0 1px #333,       /*左上*/
09               9px -9px 0 2px #333,       /*右上*/
10               -9px 9px 0 1px #333,       /*左下*/
11               9px 9px 0 1px #333         /*右下*/;
12   animation:loading 1.2s linear 0s infinite; /*无限匀速应用动画，1.2s 一次*/
13   -webkit-animation:loading 1.2s linear 0s infinite;
14   -o-animation:loading 1.2s linear 0s infinite;
15   -moz-animation:loading 1.2s linear 0s infinite;
16 }
17 @keyframes loading                          /*定义动画*/
18 {
19   from {transform:rotate(0deg);}             /*旋转 div，从 0 度开始*/
20   to {transform:rotate(360deg);}             /*旋转到 360 度*/
21 }
22 @-webkit-keyframes loading
23 {
24   from {-webkit-transform:rotate(0deg);}
25   to {-webkit-transform:rotate(360deg);}
26 }
27 @-o-keyframes loading
28 {
29   from {-o-transform:rotate(0deg);}
30   to {-o-transform:rotate(360deg);}
31 }
32 @-moz-keyframes loading
33 {
34   from {-moz-transform:rotate(0deg);}
35   to {-moz-transform:rotate(360deg);}
36 }

```

在第 4~11 行，在 div 的四周设置小黑点，第 12~15 行对 div 设置动画，该动画是第 17~36 行对应的动画，注意其中针对不同浏览器属性的不同。

注意：务必定义一个 gif 图兼容低版本浏览器。

5.6 进度条

进度条在一些网页游戏中常见，不过网页游戏中的进度条是 Flash 方面的应用。CSS 中的进度条可以应用在显示上传文件进度、显示下载文件进度、显示网页加载进度或者显示当前操作进度等场景。在 WEB QQ (<http://web.qq.com>) 页面打开时，就应用了滚动条。在 QQ 邮箱上传附件时，也应用了进度条。由于本书的侧重点在 CSS，所以本节中将只模仿进度条的实现，而非真实的进度值。在网页应用中往往使用真实的进度信息，获取进度信息的方法有硬编码、Flash 和 AJAX 等方法。图 5.14 是进度条动态变化的效果。

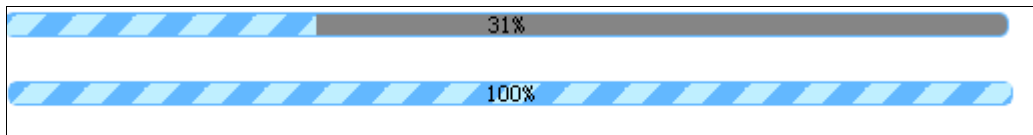


图 5.14 进度条动态变化效果

在 HTML 部分，需定义两个 div：一个外层容器；另一个是进度条，进度条与百分比数字并列写在外层容器中。HTML 代码如下：

```
01 <div id="progressbar">
02 <div id="progress"></div>
03 <p>20%</p>
04 </div>
```

然后做 CSS 的初步设置，进度条的高度和宽度由外层 div 决定，进度条 div 的高度为 100%，宽度同样使用百分比数值，这样宽度的百分比数值与真正的百分比数字是吻合的。对进度条 div 做绝对定位使之脱离文档流，然后把 z-index 值设为 -1，避免盖住百分比数字。百分比数字设为居中显示。下方为初步设置的 CSS 代码，上述设置分别对应代码的第 7~8 行、第 18~19 行、第 16 行、第 22 行以及第 11 行。其执行后的效果图如图 5.15 所示。

```
01 *{
02     margin:0;
03     padding:0;
04 }
05 #progressbar{                                /*外层容器*/
06     position:absolute;
07     height:10px;
08     width:500px;
09     border:1px solid #63B8FF;
10     border-radius:5px;
11     text-align:center;
12     line-height:10px;
13     font-size:0.8em;
14 }
15 #progress{                                    /*内层进度容器*/
16     position:absolute;                        /*绝对定位到合适位置*/
```

```

17 border:1px solid;
18 height:100%;
19 width:60%;
20 background:#BFEFFF;
21 border-radius:5px;
22 z-index:-1;
23 }

```

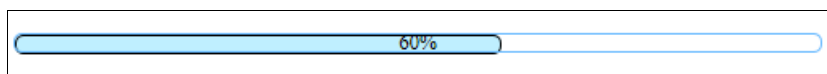


图 5.15 进度条的初步设置效果图

接下来考虑如何使进度条变化起来。需要改变的东西很明显为两处，一处是进度条的宽度，一处是百分比数字。因为在布局 CSS 的时候使用的是百分比，所以在做改变时会很简单。例如下面的这段 JavaScript 代码，可以改变一次这两处的值（仅 1 次）：

```

01 <script type="text/javascript">
02 window.onload=function(){
03   var progressbar=document.getElementById("progressbar");
04   progressbar=progressbar.getElementsByTagName("p");
05   progressbar[0].innerHTML=cent+"%";
06   var progress=document.getElementById("progress");
07   progress.style.width=cent+"%";
08 }
09 </script>

```

通过这段 JavaScript 代码，可以实现手动改变任意的值了，然而自动改变才是期望的效果。这是需要一个 setInterval 计时器反复地执行这一功能，使之不停的改变。

```

01 window.onload=function(){
02   var now=0;
03   var timer=setInterval(function(){                                /*设置定时器*/
04     if(now==100)                                                    /*进度完成后所做工作写入该循环体*/
05     {
06       clearInterval(timer);
07       window.location.href='#';
08     }
09     else
10     {
11       now+=1;                                                         /*模拟进度+1*/
12       progressfn(now);                                                /*传入改变函数*/
13     }
14   },50);
15 }
16 function progressfn(cent)                                           /*改变函数*/
17 {
18   var progressbar=document.getElementById("progressbar");
19   progressbar=progressbar.getElementsByTagName("p");

```

```

20 progressbar[0].innerHTML=cent+"%";           /*改变 HTML 文件中的文字*/
21 var progress=document.getElementById("progress");
22 progress.style.width=cent+"%";                 /*改变进度条的宽度*/
23 }

```

第4行中首先定义一个为0的变量，第5~16行是用setInterval函数做一个反复执行的功能，当变量没有达到100之前，会反复调用改变进度条的函数progressfn(cent)并使变量自增，当变量到达100时，进度条已经达到100%，这时不再进行函数。progressfn(cent)函数的功能为，变量的值通过形参cent传入函数，获取到进度条对象和百分比数值对象，通过操作CSS使进度条当前的width属性值为传进的cent数值，同时通过innerHTML使百分比数值变为传进的cent值（第20行）。

如果把class代替id，可以增强代码的重用性。最后对进度条进行美化。有几种方案可选：一是使用静态图片沿着x轴方向平铺；二是使用同态gif进度图片；三是使用CSS代码，原理同第2章中CSS3动画边框一样。本例将采用第三种方案，但不做CSS解释。执行下面的CSS代码之后即可获得动态进度条效果，完整的代码可以参考实例文档。

```

01 #progress{                                     /*对进度条应用渐变效果*/
02   background-size: 30px 30px;
03   background-image: -webkit-linear-gradient(-45deg, #63B8FF,#63B8FF 25%, #BFEFFF
04       25%,#BFEFFF 50%, #63B8FF 50%, #63B8FF 75%, #BFEFFF 75%, #BFEFFF);
05   background-image: -moz-linear-gradient(-45deg, #63B8FF,#63B8FF 25%, #BFEFFF
06       25%,#BFEFFF 50%, #63B8FF 50%, #63B8FF 75%, #BFEFFF 75%, #BFEFFF);
07   background-image: linear-gradient(-45deg, #63B8FF,#63B8FF 25%,#BFEFFF
08       25%,#BFEFFF 50%, #63B8FF 50%, #63B8FF 75%, #BFEFFF 75%, #BFEFFF);
09   -webkit-animation: animate 1.5s linear infinite;
10   -moz-animation: animate 1.5s linear infinite;
11   animation: animate 1.5s linear infinite;
12 }
13 @-webkit-keyframes animate {
14   from {
15     background-position: 0 0;
16   }
17   to {
18     background-position: 60px 30px;
19   }
20 }
21 @-moz-keyframes animate {
22   from {
23     background-position: 0 0;
24   }
25   to {
26     background-position: 60px 30px;
27   }
28 }

```

```

29 @keyframes animate {
30   from {
31     background-position: 0 0;
32   }
33   to {
34     background-position: 60px 30px;
35   }
36 }

```

注意：由于 CSS 3 动画属性对浏览器版本要求较高，如果想让低版本浏览器兼容，需使用动态 gif 图片或者使用 JavaScript 按照动画的原理兼容。

在实际应用中进度条的作用是改进用户在进行操作时的等待体验，在网页加载时应用进度条可以让用户安心地等待而不至于流失用户。一般的用户等待承受时间在 8 秒左右，用数值可以比简单的转圈效果更让用户觉得安心。实际应用中，要使用真实的数据，否则进度条的效果华而不实。另外，进度条作为网页开始前的元素，也能收到不错的效果。

5.7 图标滑动切换特效

本章的前几节是 CSS 3 中与动画有关的属性的介绍，使用 CSS 3 制作动画的好处是省去了 JavaScript 代码更加方便而且性能也很好。为了更好的应用 CSS 3 新属性，本节将只是用 CSS 3 代码制作一个图标滑动切换的特效。该特效在腾讯 SOSO 问问个人首页右侧和淘宝网便民服务栏等处均有应用。如图 5.16 所示，当鼠标移动到某一条目时，条目左侧的图标将以滑动的方式点亮。

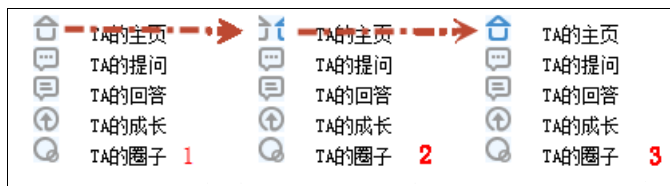


图 5.16 图标滑动切换特效

在 HTML 部分是一个无序列表，为了便于 CSS 选择，HTML 部分需要添加多个类。

```

01 <div id="div1">
02 <ul>
03   <li><span class="icon1"></span>TA 的主页</li>
04   <li><span class="icon2"></span>TA 的提问</li>
05   <li><span class="icon3"></span>TA 的回答</li>
06   <li><span class="icon4"></span>TA 的成长</li>
07   <li><span class="icon5"></span>TA 的圈子</li>
08 </ul>
09 </div>

```


在第 3~7 行，每个 span 上都应用一个类供 CSS 选择，对 li 的选择采用子类选择器：nth-child (n) 来选择。CSS 部分的代码如下：

```
01 #div1 li{list-style:none;}
02 #div1 span{height:16px;width:16px;display:inline-block;margin-right:20px;
03 transition: background-position .3s;
04 -moz-transition: background-position .3s;           /* Firefox 4 */
05 -webkit-transition: background-position .3s;        /* Safari 和 Chrome */
06 -o-transition: background-position .3s; }           /* Opera */
07 .icon1{background:url('img/1.png');background-position:0 0;}
08 .icon2{background:url('img/1.png');background-position:0 -20px;}
09 .icon3{background:url('img/1.png');background-position:0 -40px;}
10 .icon4{background:url('img/1.png');background-position:0 -117px;}
11 .icon5{background:url('img/1.png');background-position:-42px -141px;}
12 #div1 li:nth-child(1):hover .icon1{background-position:-18px 0;}
13 #div1 li:nth-child(2):hover .icon2{background-position:-18px -20px;}
14 #div1 li:nth-child(3):hover .icon3{background-position:-18px -40px;}
15 #div1 li:nth-child(4):hover .icon4{background-position:-18px -117px;}
16 #div1 li:nth-child(5):hover .icon5{background-position:-60px -141px;}
```

每个图标都是通过背景图片设置的（第 7~11 行），通过改变 background-position 的值改变背景图片的位置，切换到图标点亮的位置（第 12~16 行）。而在第 3~6 行，对 span 设置 CSS 变换，使得 span 在两种 background-position 样式之间的切换在 0.3S 内平滑切换，当鼠标移动到 li 上会触发对应的 span 的样式切换，这样一个无须任何 JavaScript 的效果就做好了。

5.8 流星划过效果

本节实例主要模仿流星划过效果。流星划过效果在百度加速乐官网上有应用，其原理为 div 绝对定位，效果图如图 5.17 所示。

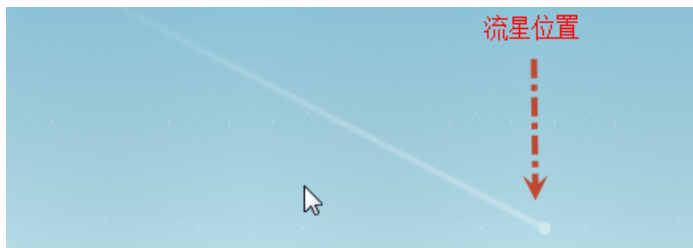


图 5.17 流星划过效果图

可以看到的流星实质是一个绝对定位的 div，它使用一张流星效果的图片作为背景。使用 JavaScript 动态地改变 div 的 left 和 top，即可改变流星的位置。布局方面的 CSS 代码为：

```
01 #container{width:100%;height:430px;border:1px solid;background:url("img/bg.png");
02 position:absolute;overflow:hidden;}
```

```

03 #star1{width:220px;height:130px;background:url("img/star.png");/*确定背景图片中星星位置*/
04     background-position:-380px 0;position:absolute;top:-150px;}
05 #star2{width:200px;height:100px;background:url("img/star.png");
06     background-position:-380px -150px;position:absolute;top:-150px;}
07 #star3{width:280px;height:150px;background:url("img/star.png");
08     background-position:-380px -270px;position:absolute;top:-150px;}

```

其中第 3~8 行是 3 个尺寸不同的流星 div，使用同一张图片和不同的定位值作为背景。

JavaScript 部分，获取到 3 个 div 之后，使用本书常用的运动框架或者 jQuery 方便地使其运动起来。

```

01 <script type="text/javascript" src="js/zQuery.js"></script>           //运动框架
02 <script type="text/javascript">
03 window.onload=function (){
04     var con=document.getElementById('container');                   //容器
05     var star=con.getElementsByTagName('div');                       //流星
06     var h=con.offsetHeight;var w=con.offsetWidth;
07     var i=0;
08     setInterval(function (){
09         var size=star[i].offsetWidth/star[0].offsetHeight;
10         var left=Math.random()*w-Math.random()*(w-star[i].offsetWidth); //随机生成定位值
11         star[i].style.left=left+'px';
12         star[i].style.top=-150+'px';
13         move(star[i],{left:[left+size*h],top:[h]});                 //使流星运动
14         i++;                                                         //不同流星切换
15         if (i==star.length)
16         {
17             i=0;                                                       //循环进行
18         }
19     },3000)
20 }
21 </script>

```

注意：如果流星部分使用 CSS 3，也可以使用纯 CSS 代码设计一个 div 并进行旋转实现。考虑到兼容性，没有使用这种做法。这里只是简单地说明流星的制作方法，在数据的计算和图像的渲染方面较为平淡。

5.9 雪花飘落效果

雪花飘落效果一般应用在冬季的网站和圣诞节时的网站背景上，与上一节中流星划过效果原理一致，都是 div 的定位。有些网站使用 JavaScript 或者 HTML 5 canvas 画布实现雪花飘落效果，本节使用 CSS 和 JavaScript（主要是定位工作），从一定程度上讲，本节的关键在雪花的实现上——使用文字实现图像效果。图 5.18 为雪花飘落效果的部分截图，实例中是全屏飘雪。

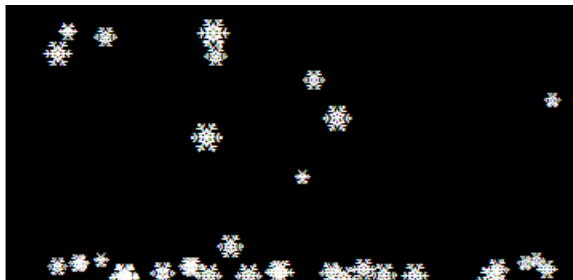


图 5.18 雪花飘落效果

本实例的设计比较巧妙之处在于使用 CSS 3 的自定义字体。在字体中，有一种图像字体，本实例中的雪花图像其实就是一个字，其大小可以使用字体大小设置。字体放置在 div 中，div 设置绝对定位。

```

01 @font-face{
02   font-family:icon;src:url("css/font/icon.ttf"),url("css/font/icon.eot");
03 }
04 body{background:black;}
05 div{
06   font-family:icon;
07   color:#fff;
08   position:absolute;
09   top:-100px;
10 }

```

在 JavaScript 部分，随机生成雪花的大小值、雪花的 left 值。然后将雪花放入新建的 div 中，并把 div 写入文档流里，然后设置定时器改变雪花的 top 定位值，当雪花落到屏幕底部时停止运动，从而创建一个雪花落地堆积的效果。JavaScript 部分的代码如下。其中获取窗口大小的函数省略了。

```

01 window.onload =function (){
02   var bd=document.getElementsByTagName('body')[0];    //获取 body
03   function snow(){                                  //雪花函数
04     var win_height=win('height');                  //获取窗口高度
05     var win_width=win('width');                     //获取窗口宽度
06     function createsnow(){                          //创建雪花函数
07       var snowdiv=document.createElement('div');   //新建一个 div
08       snowdiv.innerHTML="j";                       //写入雪花（字体的代码为 j）
09       var size=20+parseInt(Math.random()*30);       //随机生成雪花的大小
10       var left=parseInt(win_width*(Math.random()*0.98)); //随机生成 left 值
11       snowdiv.style.fontSize=size+'px';            //应用字体大小
12       snowdiv.style.width=size+'px';               //div 应用宽度
13       snowdiv.style.height=size+'px';              //div 应用高度
14       snowdiv.style.left=left+'px';                //雪花的 left 值
15       bd.appendChild(snowdiv);                     //将此雪花放入文档中
16       return snowdiv;                             //返回这个雪花对象

```

```

17 }
18 setInterval (function (){
19     var snow=createsnow();           //创建雪花并得到这一对象
20     var sbd=Math.ceil(Math.random()*3); //随机生成下落速度
21     move(snow,sbd);                   //移动雪花
22 },100);
23 function move(obj,speed){            //移动函数
24     var top=0;
25     var timer=setInterval(
26     function (){
27         top=top+speed;                //改变 top 值
28         obj.style.top=top+'px';
29         if (top>win_height-20)
30         {
31             clearInterval(timer);    //下落停止
32         }
33     },30);
34 }
35 setInterval(function(){              //定期清理，防止文档过大卡住
36 var snows=document.getElementsByTagName('div');
37 for (var i=0;i<snows.length/3 ;i++ )
38     {
39         snows[i].parentNode.removeChild(snows[i]);
40     }
41 },20000);
42 }
43 snow();                               //执行 snow 函数
44 window.onresize=function(){          //窗口改变
45     var snows=document.getElementsByTagName('div');
46     for (var i=0;i<snows.length ;i++ )
47     {
48         snows[i].parentNode.removeChild(snows[i]);
49     }
50 }
51 }

```

注意：在动画的实现中，很多都是使用定位实现运动。

5.10 数字滚动器

数字滚动器多用在展示某一产品用户数量增加的直观展示上，比如百度加速乐官网上的数字滚动效果，通常通过 AJAX 从服务器请求数据以保持数据的真实性（不过一般在产品宣传的时候有些网站也使用假数据来吸引用户）。有的时候也作为抽奖类应用的一种形

式，也有使用倒计数的某些应用。这种动态的数字展现方式，能让页面看起来不呆板。图 5.19 为本节的实例效果图，实例并没有在美化部分下工夫，主要在于实现数字滚动器这一功能。

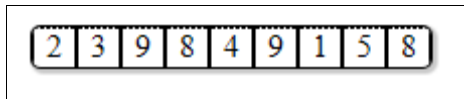


图 5.19 数字滚动器

实现的原理其实并不难，把多个带有 0~9 的列表按照左浮动的方式排列起来，然后通过 JavaScript 部分动态地改变某些列表的 top 定位值即可，外层的容器设为溢出隐藏，在外层容器不隐藏时，其真面目便会清晰的实现，一目了然，如图 5.20 所示。

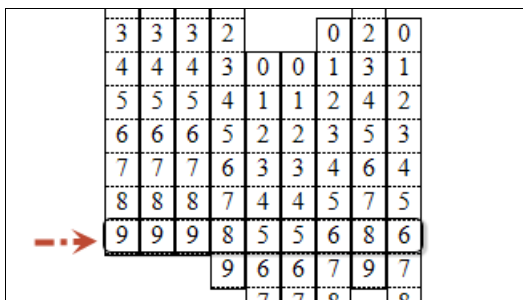


图 5.20 不隐藏时的部分图

为了使数据的总位数变得灵活可控，不能把 HTML 部分的代码固定，只在 HTML 部分写一个容器放置 JavaScript 部分动态生成的 ul。CSS 部分的代码如下，首先设定好整体的样式。

```

01  *{margin:0;padding:0;}
02  #changenumber{                                /*外层容器*/
03      border:1px solid;border-radius:5px;
04      position:absolute;
05      left:300px;top:300px;
06      box-shadow:1px 1px 2px #CCC;
07      width:auto;height:20px;overflow:hidden;    /*溢出部分隐藏*/
08  }
09  #changenumber ul{                                /*无序列表*/
10      width:20px;
11      border:1px solid;
12      text-align:center;
13      position:relative;float:left;top:0;        /*左浮动排列在一起*/
14  }
15  #changenumber li{                                /*各个列表项*/
16      height:20px;
17      border-bottom:1px dashed;
18      list-style:none;
19  }
```

由于动态改变属性值属于运动的范畴，所以需要引入与 jQuery 中 `animate` 功能相类似的 `zQuery` 来实现（主要使用缓冲运动部分）。在 JavaScript 部分创建两个函数，一个是创建函数（第 4~17 行），该函数的参数为位数 `n` 和对象，可以创建一个 `n` 位数的数字并放入对象中；另一个是改变函数（第 18~31 行），参数为数字，这一数字的位数比 `n` 小，传入数字之后便可改变 HTML 中数字的值。JavaScript 部分如下：

```

01 <script type="text/javascript" src="js/zQuery.js"></script>
02 <script type="text/javascript">
03 window.onload =function (){
04     function createnumber(n,obj){                                //创建数字函数
05         for (var i=0;i<n ;i++ )
06         {
07             var ul=document.createElement('ul');                //创建 ul
08             for (var j=0;j<10 ;j++ )
09             {
10                 var li=document.createElement('li');              //创建 li
11                 var text=document.createTextNode(j);              //创建文本节点
12                 li.appendChild(text);                              //在 li 中填入数字
13                 ul.appendChild(li);                                //将 li 放入 ul
14             }
15             obj.appendChild(ul);                                  //将 ul 放入容器
16         }
17     }
18     function change(num){                                        //改变数值函数
19         if (num%10>=0&&num>10)                                    //依次取得数字的每一位并存入数组
20         {
21             temp.push(num%10);
22             num=Math.floor(num/10);
23             change(num);
24         }else{
25             temp.push(num);
26         }
27         for (var i=0;i<temp.length ;i++ )
28         {
29             move(ul[n-1-i],{top: [-temp[i]*li_height]});          //改变某些 ul 的定位值
30         }
31     }
32     var changenumber=document.getElementById('changenumber');    //获得容器
33     createnumber(9,changenumber);                                  //在容器中创建 9 位数
34     var ul=changenumber.getElementsByTagName('ul');              //获得创建完成的每个 ul
35     var li_height=ul[0].getElementsByTagName('li')[0].offsetHeight; //获取 li 的高度供定位
36     var n=ul.length;                                              //获取位数
37     var num=1298453;var temp=[];                                  //num 为 9 位数，temp 为改变函数中的暂存数组
38     setInterval(function (){temp=[];change(num);num+=parseInt(Math.random()*1000);

```

```

39                                     //每当定时器运行时先清空 temp 的值
40 },1500);
41 }
42 </script>

```

5.11 模拟时钟

CSS 3 中增加了 2D 转换的属性,为模拟时钟的实现带来了方便。模拟时钟在网页中不常见,有些是 Flash 应用。Web QQ 上有一款模拟时钟,对于 Web 上的应用是很实用的。在本节中将一步步地实现图 5.21 所示的效果。

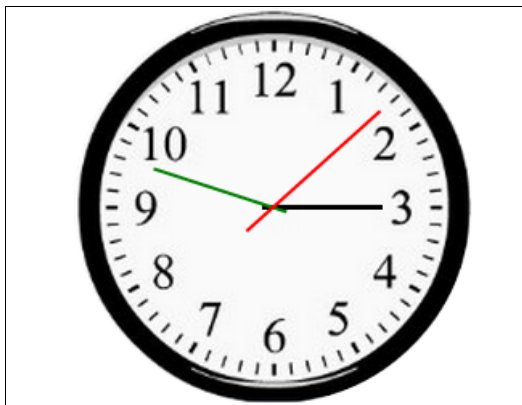


图 5.21 模拟时钟

本实例仅使用一张背景图片取代了各种样式的制作,所以在 HTML 部分非常简单——一个主容器、一个时针容器、一个分针容器和一个秒针容器。

```

01 <div id="clock">
02 <!--<ul id="wq">12 个刻度 li,背景图片中有刻度或不需要刻度时可不写</ul>-->
03 <div id="hour"></div>
04 <div id="minute"></div>
05 <div id="second"></div>
06 </div>

```

本实例实现过程中会确定每个刻度的位置,以便说明刻度的实现过程,但因本实例最终使用的背景图片中带有刻度,所以最终刻度是隐藏的、JavaScript 与刻度有关的实现部分也可以省略。执行上面的 HTML 只能看到 12 个列表项目符号(12 个黑点)。为了增强代码的通用性,即始终可以调整大小以适应各种浏览器窗口,在元素的大小和定位时使用百分比代替像素值,并将可以动态改变和使用到 CSS 3 2D 转换的部分用 JavaScript 动态计算。

首先获取到所有需要使用到的元素,从时钟大小入手,先用 JavaScript 获取到可视区的宽度和高度,比较两数取得最小数作为时钟的宽度。同时将此数的一半作为圆角的参数,即可得到一个适合浏览器窗口大小的圆形。

```

01 window.onload=function (){

```

```

02 var clock_margin=5; //设置 clock 的 margin 像素值
03 var view_width=document.documentElement.clientWidth; //获取可视区宽度
04 var view_height=document.documentElement.clientHeight; //获取可视区高度
05 var clock=document.getElementById("clock"); //获取时钟
06 var lis=document.getElementById("lis"); //获取刻度 ul
07 var lis_li=lis.getElementsByTagName("li"); //获取每个刻度 li
08 var hour=document.getElementById("hour"); //获取时钟
09 var minute=document.getElementById("minute"); //获取分针
10 var second=document.getElementById("second"); //获取秒针
11 var clock_width=
12 (view_width<=view_height)?view_width-2*clock_margin:view_height-2*clock_margin;
13 //时钟的宽和高与可视区最小长度相等
14 clock.style.margin=clock_margin+"px auto"; //clock 外边距与居中
15 clock.style.height=clock_width+"px";
16 clock.style.width=clock_width+"px";
17 var clock_radius=Math.ceil(clock_width/2); //圆角
18 clock.style.borderRadius=clock_radius+"px";

```

下面考虑刻度值，在 CSS 3 2D 转换 (transform) 中，使用 translate 可以改变元素的位置，使用 rotate 可以旋转元素。在刻度上需要使用。首先在 CSS 中对刻度值做如下设置：

```

01 #lis li{
02     display:block;
03     position:absolute;
04     border:1px solid;
05     background:black;
06 }

```

在 JavaScript 依据窗口大小动态计算刻度的合适大小：

```

01 var li_width=Math.ceil(clock_radius*0.02); //计算 li 宽度
02 var li_height=Math.ceil(clock_radius*0.2); //计算 li 长度

```

动态确定刻度值所在的方位，以数字 2 所在的刻度为例，根据三角函数和加减法，translate 的参数值和 rotate 的参数值可以由图 5.22 所示的方法求得。

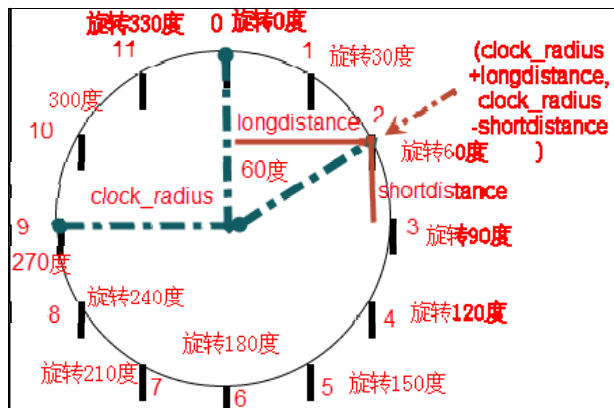


图 5.22 刻度求值方法

这种情况下使用 for 循环和 switch 语句是最合适的，JavaScript 代码如下，计算完成后将值写入 CSS：

```

01 var longdistance=Math.ceil(clock_radius*0.866);//计算 li 的 translate 时需要的较长宽度
02 var shortdistance=Math.ceil(clock_radius*0.5);      //计算 li 的 translate 时需要的较短宽度
03 for (var i=0;i< lis_li.length;i++ )
04 {
05     var distancestr,var degstr;                      //参数存储变量
06     switch(i){
07     case 0:distancestr="("+clock_radius+"px,0)";degstr="(0deg);";break;
08     case 1:distancestr="("+clock_radius+shortdistance)+"px,"
09         +(clock_radius-longdistance)+"px";degstr="(30deg);";break;
10     case 2:distancestr="("+clock_radius+longdistance)+"px,"
11         +(clock_radius-shortdistance)+"px";degstr="(60deg);";break;
12     case 3:distancestr="("+clock_width+"px,"
13         +clock_radius+"px";degstr="(90deg);";break;
14     case 4:distancestr="("+clock_radius+longdistance)+"px,"
15         +(clock_radius+shortdistance)+"px";degstr="(120deg);";break;
16     case 5:distancestr="("+clock_radius+shortdistance)+"px,"
17         +(clock_radius+longdistance)+"px";degstr="(150deg);";break;
18     case 6:distancestr="("+clock_radius+"px,"
19         +clock_width+"px";degstr="(180deg);";break;
20     case 7:distancestr="("+clock_radius-shortdistance)+"px,"
21         +(clock_radius+longdistance)+"px";degstr="(210deg);";break;
22     case 8:distancestr="("+clock_radius-longdistance)+"px,"
23         +(clock_radius+shortdistance)+"px";degstr="(240deg);";break;
24     case 9:distancestr="("+0+"px,"+clock_radius+"px";degstr="(270deg);";break;
25     case 10:distancestr="("+clock_radius-longdistance)+"px,"
26         +(clock_radius-shortdistance)+"px";degstr="(300deg);";break;
27     case 11:distancestr="("+clock_radius-shortdistance)+"px,"
28         +(clock_radius-longdistance)+"px";degstr="(330deg);";break;
29     default:distancestr="(0,0)";degstr="(0deg);";break;
30     }
31     lis_li[i].style.cssText="-webkit-transform:translate"+distancestr+"rotate"+degstr
32         +"-moz-transform:translate"+distancestr+"rotate"+degstr
33         +"-o-transform:translate"+distancestr+"rotate"+degstr
34         +"-ms-transform:translate"+distancestr+"rotate"+degstr
35         +"transform:translate"+distancestr+"rotate"+degstr; //写入 CSS
36     lis_li[i].style.width=li_width+"px";lis_li[i].style.height=li_height+"px";    //刻度的大小
37 }

```

按照这段 JavaScript 代码执行之后的效果如图 5.23 中左图所示。可以看到某些刻度混乱，这是因为 CSS 中没有设置 2D 转换时绕哪一点旋转的缘故。需在 li 的 CSS 中添加下面一行代码，即可解决问题，解决后的效果如图 5.23 中右图所示，到这里表盘和表盘刻度制作完成。

```

01 -webkit-transform-origin:0% 0%;

```

```

02 -moz-transform-origin:0% 0%;
03 -o-transform-origin:0% 0%;
04 -ms-transform-origin:0% 0%;
05 transform-origin:0% 0%;

```

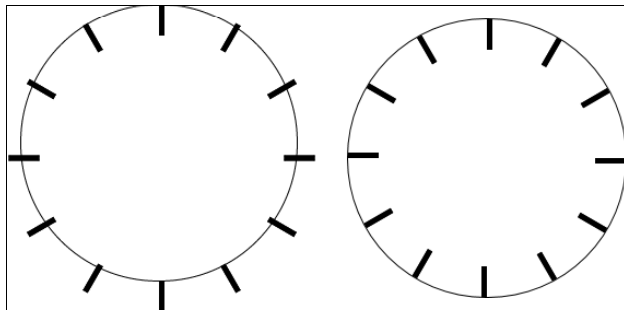


图 5.23 修改效果

在钟表指针大小和定位的控制上，使用可视区宽度计算比使用百分比计算更精准，所以在 CSS 部分之规定确定的属性，另外希望指针绕其中部的某个位置旋转，而不是顶端，所以使用 `transform-origin` 设置，这一属性也将影响到定位的计算。（`transform-origin`：宽度方向的百分比、高度方向的百分比。）

```

01 #hour{                                     /*时针*/
02   background:black;
03   position:absolute;z-index:1;
04   -webkit-transform-origin:50% 10%;        /*设置变换基准位置*/
05   -moz-transform-origin:50% 10%;
06   -o-transform-origin:50% 10%;
07   -ms-transform-origin:50% 10%;
08   transform-origin:50% 10%;
09 }
10 #minute{                                   /*分针*/
11   background:green;
12   position:absolute;z-index:2;
13   -webkit-transform-origin:50% 10%;        /*设置变换基准位置*/
14   -moz-transform-origin:50% 10%;
15   -o-transform-origin:50% 10%;
16   -ms-transform-origin:50% 10%;
17   transform-origin:50% 10%;
18 }
19 #second{                                   /*秒针*/
20   background:red;
21   position:absolute;z-index:3;
22   -webkit-transform-origin:50% 20%;        /*设置变换基准位置*/
23   -moz-transform-origin:50% 20%;
24   -o-transform-origin:50% 20%;
25   -ms-transform-origin:50% 20%;

```

```

26 transform-origin:50% 20%;
27 }

```

计算指针的大小及定位，获取当前时间，通过当前时间计算每个指针的旋转度数（因初始状态下 3 个指针都指向数字 6 的位置堆叠在一起，所以在计算旋转度数时应补充 180 度才能得到正确数值），最后将所有属性应用到 CSS，效果图如图 5.24 所示。

```

01 var hours;var minutes;var seconds;           //时间变量
02 var hour_height=0.3;var hour_width=0.01;     //指针宽度百分比参数
03 var hour_left;var hour_top;
04 var minute_height=0.35;var minute_width=0.007;var minute_left;var minute_top;
05 var second_height=0.45;var second_width=0.005;var second_left;var second_top;
06 hour_height=Math.floor(hour_height*clock_width); //指针的大小及定位属性值计算
07 hour_width=Math.floor(hour_width*clock_width);
08 hour_left=Math.floor(clock_radius-hour_width*0.5);
09 hour_top=Math.floor(clock_radius-hour_height*0.1); //小数部分为 transform-origin 中的数值
10 minute_height=Math.floor(minute_height*clock_width);
11 minute_width=Math.floor(minute_width*clock_width);
12 minute_left=Math.floor(clock_radius-minute_width*0.5);
13 minute_top=Math.floor(clock_radius-minute_height*0.1);
14 second_height=Math.floor(second_height*clock_width);
15 second_width=Math.floor(second_width*clock_width);
16 second_left=Math.floor(clock_radius-second_width*0.5);
17 second_top=Math.floor(clock_radius-second_height*0.2);
18 var hour_css_text="width:"+hour_width+"px;height:"+hour_height+"px;left:"
19   +hour_left+"px;top:"+hour_top+"px;"; //css 属性文本，将于下方与旋转值同时应用
20 var minute_css_text="width:"+minute_width+"px;height:"+minute_height+"px;left:"
21   +minute_left+"px;top:"+minute_top+"px;";
22 var second_css_text="width:"+second_width+"px;height:"+second_height+"px;left:"
23   +second_left+"px;top:"+second_top+"px;";
24 hours = new Date().getHours();minutes = new Date().getMinutes();
25 seconds = new Date().getSeconds();           //获取时间
26 hours_deg="("+hours/12*360+180)+"deg;"; //计算旋转度数，注意补充差值 180 度
27 minutes_deg="("+minutes/60*360+180)+"deg;";
28 seconds_deg="("+seconds/60*360+180)+"deg;";
29 hour.style.cssText=hour_css_text+"-moz-transform:rotate"+hours_deg+"-webkit-transform:rotate"+
   hours_deg+"-o-transform:rotate"+hours_deg+"-ms-transform:rotate"+hours_deg+"transform:r
   otate"+hours_deg; //将大小、定位、旋转同时应用到指针
30 minute.style.cssText=minute_css_text+"-moz-transform:rotate"+minutes_deg+"-webkit-transform:
   rotate"+minutes_deg+"-o-transform:rotate"+minutes_deg+"-ms-transform:rotate"+minutes_de
   g+"transform:rotate"+minutes_deg;
31 second.style.cssText=second_css_text+"-moz-transform:rotate"+seconds_deg+"-webkit-transform:
   rotate"+seconds_deg+"-o-transform:rotate"+seconds_deg+"-ms-transform:rotate"+seconds_d
   eg+"transform:rotate"+seconds_deg;

```

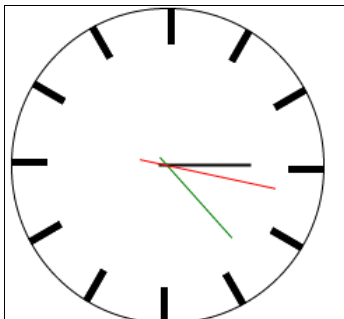


图 5.24 设置指针

到这一步，一个纯 CSS 时钟就完成了，可以手动刷新模拟指针运动。剩余的工作是：把指针的设置过程封装为函数，放在定时器里执行这一函数，当有窗口大小改变时重新执行钟表设置函数即可。

```
01 window.onload=function (){
02   clock();
03   var t=setInterval(function (){clock();},1000);
04   window.onresize = function(){clock();}
05 }
```

如果想增强钟表的样式，可以选择背景图片，并运用 `background-size` 属性改变大小。如果背景图片中存在刻度，可以将 HTML 中的刻度列表不显示。时钟构造函数在时钟大小的设置处添加：

```
01 clock.style.backgroundSize=clock_width+"px "+"clock_width+"px";
```

在 CSS 中添加背景图片路径，同时将刻度隐藏：

```
01 #clock{background:url(img/clock.jpg) no-repeat;}
02 #lis li{display:none;}
```

即可实现。使用背景图片的效果如图 5.20 所示。这样更换钟表的样式就会变得方便快捷，仅需改变背景图片。至此模拟时钟的实现过程结束，完整的代码可以参考实例。

5.12 苹果著名的 DOCK 栏

本节要做的效果是 DOCK 栏，那么什么是 DOCK 栏？见过或用过苹果操作系统的人都知道位于桌面底部有一排图标，Android 手机和 iPhone 的桌面下方也有一排固定的图标，那就是 DOCK 栏。如果网页设计者想制作一个仿操作系统的页面，那么这个 DOCK 栏一定是个不错的选择。图 5.25 是本节中 DOCK 栏的效果图，是仿苹果操作系统 DOCK 栏的效果，在鼠标指向图标时具有动画的效果。

最终目标明确之后，首先需要考虑 HTML 部分怎么写。在这种场景下，当鼠标单击图标时，往往是实现某部分 JavaScript 代码或者跳转到某个链接，所以每个图标都需要一个 `a` 标签和一个 `img` 标签组合起来，就像这样：

```
<a href="javascript:show('player');"></a>
```



图 5.25 DOCK 栏

在定位方面，内层一排图标最好不要选择 span 还是 div。最外层是少不了 div 的，但是内层的这些图标，为了实现方便和动态删减，最好还是用一个无序列表 ul 包装每个 a 标签，然后整体做定位，这样需要添加或者减少的时候，只需要对 a 标签操作即可。还有一个 DOCK 背景，这样整个 HTML 部分应该是下面这样：

```

01 <div id="bottom_dock">
02   <a href=""></a>
03   <a href=""></a>
04   <a href=""></a>
05   <a href="javascript:show('browser');"></a>
06   <a href=""></a>
07   <a href=""></a>
08   <a href=""></a>
09   <a href="javascript:show('player');"></a>
10   <a href=""></a>
11   <a href=""></a>
12   <div id="bottom_dock_background"></div>
13 </div>

```

只有 HTML 代码的执行效果如图 5.26 所示（该图片被整体缩小过）。



图 5.26 纯 HTML 执行效果

可以看到没有 CSS 代码画面的效果会很差。首先对图标做一下初步定位布局。

```

01 #bottom_dock                                /*DOCK 容器*/
02 {
03   height:auto;
04   width:auto;
05   position:absolute;
06   bottom:0;
07   overflow:hidden;

```

```

08 }
09 #bottom_dock_background      /*DOCK 背景*/
10 {
11     position:absolute;
12     bottom:0;
13     width:auto;
14     z-index:-1;
15 }
16 #bottom_dock>a>img          /*DOCK 图标*/
17 {
18     border:0;
19     width:64px;
20     height:64px;
21 }

```

第 1~8 行对整个 DOCK 块的位置定位，一般 DOCK 栏是需要放在底部居中的位置，为了使居中能够自适应不同的浏览器以及不同的情况，建议把居中的定位用 JavaScript 来实现。这里仅设置整个 div 于底部，高度宽度自动，超出部分隐藏。9~15 行对 DOCK 背景做定位，第 13 行能够为以后 JavaScript 部分动画时 DOCK 背景的宽度自动改变做铺垫。第 14 行使背景位于图标的下方。第 16~21 行对每个图标的大小进行设置，为了方便讲解，大小部分使用像素值，这样将不能自适应各种不同的情况，实际使用时需要根据场景来计算合适的大小值，或者可以使用百分比来设置。初步定位以后的效果如图 5.27 所示。

注意：第 18 行如果不写，在 IE9 浏览器下图标将带有边框，这里图标不需要边框，所以这行代码在 IE9 下是需要的。



图 5.27 初步定位效果

到这一步与最终的效果已经很接近了，只是缺少了动画效果。实现动画效果可以使用 CSS 3 中的动画，但 CSS 3 动画对 IE 浏览器的要求需要 IE10 版本及以上，为了使 IE9 也能实现，动画实现部分选择 JavaScript 来实现，这里笔者只说明 JavaScript 部分需要实现的效果，代码部分请读者自行理解。

JavaScript 部分需要做两个方面的工作，首先是对 CSS 部分预留的居中定位的实现，其次是对鼠标悬浮在图标上带来的图标大小的变化。这里的变化可以用图 5.28 直观地说明。

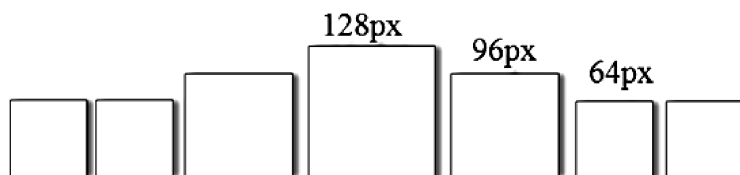


图 5.28 大小示意图

当鼠标悬停在某个图标时，通过 JavaScript 使当前的图标的长度和宽度由开始的 64 像素变为 128 像素，两侧图标变大的程度稍小，变为 96 像素。在鼠标移开当前元素时，复原原来的大小。图 5.29 所示的情况需要注意，在 JavaScript 中需要特别考虑这种情况。

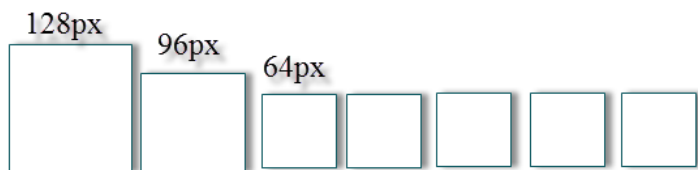


图 5.29 特殊情况

下面是实现这个功能的一种 jQuery 代码，添加这段代码并且引入 jQuery 之后就可以实现 DOCK 栏的动画效果了。

```

01 $(function(){                                     //调整 DOCK 栏居中
02   var n=0;
03   var window_width=$(document.body).width();
04   var dock_width=$("#bottom_dock").width();
05   var left=parseInt((window_width-dock_width)/2);
06   $("#bottom_dock").css("left",left);
07   $("#bottom_dock>a").hover(                       //鼠标悬浮事件
08     function(){
09       n=$(this).index();
10       dock_big(n);
11       if(0==n)
12       {
13         dock_big_lite(n+1);
14       }
15       else if($("#bottom_dock>a").length==n)
16       {
17         dock_big_lite(n-1);
18       }
19       else
20       {
21         dock_big_lite(n-1);
22         dock_big_lite(n+1);
23       }
24     },
25     function(){                                     //鼠标离开
26       dock_small(n);
27       if(0==n)
28       {
29         dock_small(n+1);
30       }
31       else if($("#bottom_dock>a").length==n)
32       {
33         dock_small(n-1);

```

```

34     }
35     else
36     {
37         dock_small(n-1);
38         dock_small(n+1);
39     }
40 }
41 );
42 //结束 hover
43 function dock_big(index){
44     //放大缩小函数
45     $("#bottom_dock>a>img").eq(index).stop(true,false).animate(
46     {"width":"128px","height":"128px"},"fast","swing" );
47     $("#bottom_dock").stop(true,false).animate({"left":left-46},"fast","swing");
48 }
49 function dock_small(index){
50     $("#bottom_dock>a>img").eq(index).stop(true,false).animate
51     ({ "width":"64px","height":"64px"},"fast","swing");
52     $("#bottom_dock").stop(true,false).animate({"left":left},"fast","swing");
53 }
54 function dock_big_lite(index){
55     $("#bottom_dock>a>img").eq(index).stop(true,false).animate(
56     {"width":"96px","height":"96px"},"fast","swing" );
57 }
58 }
59 })

```

5.13 苹果系统的 Stack 特效

苹果公司的产品总是在用户体验上表现出色，不论是手机系统还是电脑系统，都以美观著称。在 iMac 上有两款很吸引眼球的应用，一个是底部的 DOCK 栏，另一个是本节实例的 stack。在 iMac 上，Stack 是一个放置软件的抽屉，它有两个鲜明的特点，一是弹出效果，二是弯曲效果，如图 5.30 所示（由于篇幅原因该图没有完全显示）。

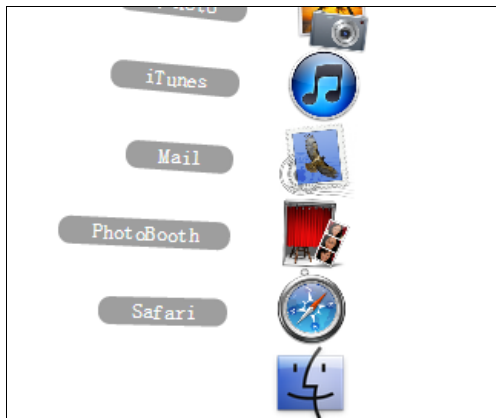


图 5.30 苹果系统 Stack

从代码上讲，CSS 3 2D 变换起到了重要的作用，因为 Stack 的两个特点正好对应 2D 变换中的平移和旋转。在 HTML 部分需要的是一个无序列表，它的结构是这样的：

```
<div id="stack">
<ul>
<li></li>           //多个 li
```

使用 CSS 3 可以很轻易地用 transform 属性把有限个图标按照固定的数值平移一定的距离和旋转一定的度数，而对于数量不固定的图标来说，却并不是一件容易的事，这同时使得 CSS 3 动画难以开展，在本实例中最终选择的是使用 JavaScript 对图标做平移与旋转变换，CSS 的主要工作是整体与部分的定位。

```
01  *{margin:0;padding:0;}
02  #stack{
03      position:absolute;left:300px;bottom:64px;           //整体脱离文档流，Finder 图标位于页面下部
04  }
05  #stack>ul>li{list-style:none;position:absolute;
06      -webkit-transform:rotate(0deg);                     //这里只是为 JavaScript 改变属性的值做基础
07      -moz-transform:rotate(0deg);
08      -ms-transform:rotate(0deg);
09      -o-transform:rotate(0deg);
10      transform:rotate(0deg);}
11  #stack>ul>li>img{width:50px;height:50px;}             //图标的尺寸设置
12  #stack>ul>li>span{                                     //通过 JavaScript 获取 alt 属性作为图标的说明
13      padding-left:20px;padding-right:20px;width:auto;border:1px solid
14      #9E9E9E;border-radius:8px;                           //文字高度自适应、左右预留一部分空白更协调
15      background:#9E9E9E;color:#FFFFFF;
16      font-size:14px;text-align:center;                   //文本居中
17      display:block;position:absolute;top:40%;right:80px;}//绝对定位使其位于图标的左边
```

JavaScript 部分的代码看似也很简单，只不过对于图标以缓冲方式运动到合理位置部分的工作写在了名为 zQuery 的运动框架中，实际上整个代码并不是简单的。JavaScript 部分如下：

```
01  <script type="text/javascript" src="js/zQuery.js"></script>
02  <script type="text/javascript">
03  window.onload =function (){
04      var stack=document.getElementById('stack');         //获取 stack
05      var stack_ul=stack.getElementsByTagName('ul')[0];    //获取 ul
06      var stack_ul_lis=stack_ul.getElementsByTagName('li'); //获取所有 li
07      var li_height=stack_ul_lis[0].offsetHeight;         //获取每个图标高度
08      var R=40*li_height;                                  //通过图标高度设置一个合理的圆半径，做转换使用
09      for (var i=0;i<stack_ul_lis.length-1 ;i++ )        //遍历 li
10      {
11          var img=stack_ul_lis[i].getElementsByTagName('img')[0]; //获取到当前的 li
12          var text=document.createTextNode(img.getAttribute('alt'));
13          //创建一个文本节点，该节点的值为图片的 alt 属性
```

```

14     var span=document.createElement('span');           //创建一个 span
15     span.appendChild(text);                             //将文本节点追加到 span
16     stack_ul_lis[i].appendChild(span);                 //将 span 追加到当前的 li 中
17     stack_ul_lis[i].style.opacity='0';                 //在初始状态下隐藏图标
18 }
19 var btn=false;                                         //判断单次点击与双击点击
20 stack_ul_lis[stack_ul_lis.length-1].onclick=function (){
21     btn=!btn;                                           //点击变量取反
22     btn?show_stack():hide_stack();                     //奇数次点击展开、偶数次隐藏
23 }
24 function show_stack(){//展示函数
25     for (var i=0;i<stack_ul_lis.length-1 ;i++ )      //遍历 li
26     {
27         var deg=1.3*(stack_ul_lis.length-i-1)/180*Math.PI; //根据设置的半径对每个图标做
合适的转换
28         var tl=Math.ceil(R*(1-Math.cos(deg)));
29         var tt=Math.ceil(R*Math.sin(deg));             //translate 参数
30         var d1=Math.cos(deg);                          //角度参数
31         var d2=Math.sin(deg);
32         var d3=-Math.sin(deg);
33         var d4=Math.cos(deg);
34         move(stack_ul_lis[i],{transform:[d1,d2,d3,d4,tl,-tt],opacity:[100]}); //使用 matrix 对图
标做转换
35     }
36 }
37 function hide_stack(){//隐藏函数
38     for (var i=0;i<stack_ul_lis.length-1 ;i++ )      //遍历 li
39     {
40         move(stack_ul_lis[i],{transform:[1,0,0,1,0,0],opacity:[0]}); //透明度为 0，位置还原
41     }
42 }
43 }
44 </script>

```

在代码的第一行首先引入运动框架，如有问题可参考该运动框架的函数使用说明章节。在获取到需要使用的元素后，在第 7~8 行通过设置一个半径便于对图标做圆弧分布处理。在第 9~16 行中，通过 JavaScript HTML DOM 操作，把每一个图标的 alt 属性提取出来放在新建的文本节点中，并把该节点放入文本说明 span 中，span 的样式已经在 CSS 中定义过，同时把这些图标隐藏起来。

第 19~23 行，通过开关 btn 的控制，实现奇数次点击展开，偶数次点击隐藏。

在第 24~38 行的展示函数中，通过 i 确定该图标旋转的合适角度，计算平移变换的合适参数，然后使用 move 函数将该图标转换到合理的位置，同时变为不透明。在该处的 move 函数中，使用的是 2D 混合变换的 matrix 函数，旋转和平移同时使用时需要六个参数，分别是角度的余弦值、角度的正弦值、角度正弦值的负数、角度的余弦值、平移到的 X 坐标

和平移到的 Y 坐标（实现的原理是矩阵变换，这里不解释）。在 `move` 函数里，用到一些复杂的知识，如 `Math` 函数、正则表达式、多重循环等，无法解释，只要会使用该函数即可（使用方法在其他章节）。

在第 39~46 行的隐藏函数中，将所有的图标还原到未转换之前的状态并设置为透明。

注意：在本实例的运动框架函数中，传入的参数是“buffer”（即缓冲运动），使用最新的 zQuery 框架也可以实现“flex”（即弹性运动）。

5.14 扇形展开

使用 CSS 3 做出的网页效果一定能给每个用户眼前一亮的感觉。本节是使用 CSS 3 新属性制作的扇形展开效果，效果图如图 5.31 所示（图中的数值和箭头是为后文设定的，不属于效果本身）。主要功能有：当页面加载完成，所有的卡片就会按照图中所示的效果自动展开；当点击封面时，展开或折叠所有卡片；当点击封面之外的任何一张时，被点击的卡片会旋转到中部（整体旋转），并且这张卡片上的文字展示出来。



图 5.31 扇形展开效果

虽然本实例的功能看似简单，但是代码较为综合，使用到的属性较多，尤其是 CSS 3 中的属性。使用本实例可以使 CSS 3 的属性理解更加深入。首先是 HTML 部分的结构：

```
01 <div id="sb-container" class="sb-container">
02 <div> <span class="sb-icon icon-cog"></span>
03 <h4><span>设置</span></h4>
04 </div>
05 多个 div
06 <div>
07 <h4><span>点击展开/折叠</span></h4>
08 <h5><span> &hearts; COLORFUL</span></h5>
09 </div>
10 </div>
```

HTML 的第 1 行中的 div 对应整体,第 2 行的 div 对应一张卡片,此 div 中的第一个 span 对应图中 1 处的图标(此处的图标并非图片,而是字体),第二个 span 对应卡片上的文字。第 8 行对应图中 4 处的字体。在 CSS 部分,主要使用自定义字体、2D 转换、动画、阴影和渐变等 CSS 3 属性,运行代码需要标准化浏览器的支持。因 CSS 部分代码较多,所以省略了某些对字体的设置,并且分为几个部分说明。

对整体的设置如下,第 2 行的相对定位使得整体成为每张卡片的父元素。

```
01 .sb-container{                               /*外层容器*/
02   position:relative;
03   margin:30px auto;width:500px;
04   text-align: center;
05 }
```

对所有卡片进行通用设置,主要包括尺寸、定位、背景色、指针类型、第 5~9 行的旋转基点设置(使卡片绕图中 3 处的位置旋转)以及第 10~13 行对卡片旋转指定动画效果,动画效果在 0.5s 内匀速完成。

```
01 .sb-container div{
02   width: 130px;height: 400px;
03   position: absolute;top:0;left:0;
04   background: #fff;cursor:pointer;
05   -webkit-transform-origin: 25% 90%;           /*设置转换基点*/
06   -moz-transform-origin: 25% 90%;
07   -o-transform-origin: 25% 90%;
08   -ms-transform-origin: 25% 90%;
09   transform-origin: 25% 90%;
10   -webkit-transition:-webkit-transform .5s ease; /*CSS 过渡*/
11   -moz-transition:-moz-transform .5s ease;
12   -o-transition:-o-transform .5s ease;
13   transition:transform .5s ease;
14 }
```

使用 nth-child 选择器对封面之外的卡片做个性化设置,包括卡片颜色和卡片内外阴影。内阴影可以制作高光效果。如果把每张卡片的旋转属性写在这里会造成代码非常多,但是用途不多,对于这一部分相似的代码采用 JavaScript 生成即可。

```
01 .sb-container div:nth-child(1){
02   background-color: #ea2a29;                   /*颜色设置*/
03   box-shadow: -1px -1px 3px rgba(0,0,0,0.1), 1px 1px 1px rgba(0,0,0,0.1),
04               inset 0 3px 0 rgba(255, 255, 255, 0.2); /*阴影设置*/
05 }
06 .sb-container div:nth-child(2){
07   background-color: #f16729;                   /*因各部分不同,没有合并*/
08   box-shadow: -1px -1px 3px rgba(0,0,0,0.1), 2px 2px 1px rgba(0,0,0,0.1),
09               inset 0 3px 0 rgba(255, 255, 255, 0.2);
10 }
```

```

11                                     /*此处省略中间部分*/
12 .sb-container div:nth-child(11){
13   background-color: #ca0d86;
14   box-shadow: -1px -1px 3px rgba(0,0,0,0.1), 11px 11px 18px rgba(0,0,0,0.4),
15               inset 0 3px 0 rgba(255, 255, 255, 0.2);
16 }

```

对卡片的封面（也就是最后一张卡片）进行设置，第2行中为封面设置背景图，第3~6行对其设置多个阴影。使用 `after` 在封面上生成一个装订按钮，该按钮为使用 `border-radius` 属性制作的圆形，在第13~21行使用了线性渐变增强它的立体感。

```

01 .sb-container div:last-child{                                     /*封面设置*/
02   background: #645b5c url(images/cover.jpg) repeat center center;
03   box-shadow:
04     -1px -1px 3px rgba(0,0,0,0.2),
05     12px 12px 20px rgba(0,0,0,0.6),
06     inset 2px 2px 0 rgba(255, 255, 255, 0.1);
07 }
08 .sb-container div:last-child:after{
09   content: "";
10   position: absolute;bottom: 15px;left: 15px;
11   width: 20px;height: 20px;border-radius: 50%;
12   background: #dddddd;
13   background: -moz-linear-gradient(-45deg, #dddddd 0%, #58535e 48%, #889396 100%);
14   background: -webkit-gradient(linear, left top, right bottom,           /*封面渐变设置*/
15     color-stop(0%,#dddddd), color-stop(48%,#58535e), color-stop(100%,#889396));
16   background: -webkit-linear-gradient(-45deg, #dddddd 0%,#58535e 48%,#889396
17     100%);
18   background: -o-linear-gradient(-45deg, #dddddd 0%,#58535e 48%,#889396 100%);
19   background: -ms-linear-gradient(-45deg, #dddddd 0%,#58535e 48%,#889396 100%);
20   background: linear-gradient(135deg, #dddddd 0%,#58535e 48%,#889396 100%);
21   filter: progid:DXImageTransform.Microsoft.gradient(
22     startColorstr='#dddddd', endColorstr='#889396',GradientType=1 );
23   box-shadow: -1px -1px 1px rgba(0,0,0,0.5), 1px 1px 1px rgba(255,255,255,0.1);
24 }
25 .sb-container div:last-child h5{
26   font-size: 40px;
27   white-space: nowrap;                                           /*文本不换行*/
28   position: absolute; top: 0px;left: 0px;
29   line-height: 40px;
30   color: #111;
31   text-shadow: -1px -1px 1px rgba(255,255,255,0.1);
32   -webkit-transform: rotate(-90deg) translateX(-157%) translateY(73px); /*2D 变换*/
33   -moz-transform: rotate(-90deg) translateX(-157%) translateY(73px); /*旋转文字*/
34   -o-transform: rotate(-90deg) translateX(-157%) translateY(73px);   /*移动文字*/

```

```

35 -ms-transform: rotate(-90deg) translateX(-157%) translateY(73px);
36 transform: rotate(-90deg) translateX(-157%) translateY(73px);
37 -webkit-transform-origin: 0 0; /*变换基点设置*/
38 -moz-transform-origin: 0 0;
39 -o-transform-origin: 0 0;
40 -ms-transform-origin: 0 0;
41 transform-origin: 0 0;
42 -webkit-touch-callout: none; /*不允许选中文本*/
43 -webkit-user-select: none;
44 -khtml-user-select: none;
45 -moz-user-select: none;
46 -ms-user-select: none;
47 user-select: none;
48 }

```

如果不使用第 27 行，4 处的文字会显示两行。第 31~36 行对文字旋转和平移到合适的位置，变换的基点有第 37~41 行的代码设置，第 42~47 行是 CSS 3 中一个不常见的属性，它使得用户不能选中文字以便保持最佳的展示效果。

接下来是图中 1 位置的图标，这里使用的图标是自定义字体中的文字符号，既节省图片又节省代码，使用 before 伪对象实现，第 14~24 行 content 属性的值对应字体的编号。

```

01 @font-face{
02   font-family:'icon';
03   src:url("font/icons.eot"),url("font/icons.svg"),url("font/icons.ttf"),url("font/icons.woff");
04 }
05 span.sb-icon:before {
06   font-family: 'icon';font-style: normal;font-weight: normal;font-size: 60px;
07   display: block;
08   text-decoration: inherit;text-align: center;
09   text-shadow: 1px 1px 1px rgba(127, 127, 127, 0.3),0 0 1px #000;
10   line-height: 64px;
11   width: 100%;
12   color: #000;
13 }
14 .icon-cog:before { content: '\35'; } /* '5' */
15 .icon-flight:before { content: '\37'; } /* '7' */
16 .icon-eye:before { content: '\34'; } /* '4' */
17 .icon-install:before { content: '\39'; } /* '9' */
18 .icon-bag:before { content: '\36'; } /* '6' */
19 .icon-globe:before { content: '\38'; } /* '8' */
20 .icon-picture:before { content: '\32'; } /* '2' */
21 .icon-video:before { content: '\30'; } /* '0' */
22 .icon-download:before { content: '\41'; } /* 'A' */
23 .icon-mobile:before { content: '\42'; } /* 'B' */
24 .icon-camera:before { content: '\33'; } /* '3' */

```

由于动画工作由 CSS 3 中的 transition 属性完成，所以 JavaScript 部分比较简单，只需改变旋转度数即可，不需要在度数的渐变上考虑任何东西。关键在于各个卡片的旋转度数如何计算的问题。下面的这段 JavaScript 代码是一种可以实现的解决方案，与 jQuery 相比，此代码更加轻量：

```

01 function index(current, obj){                                //获取元素索引值
02     for (var i = 0; i < obj.length; i++){
03         if (obj[i] == current) {return i;} }
04 window.onload =function (){
05     var container=document.getElementById('sb-container');    //获取整体
06     var cards=container.getElementsByTagName('div');          //获取卡片
07     var n=cards.length;var deg=180/n;var click=0; //卡片数量、相邻卡片旋转度数差、点击次数
08     for (var i=0;i<n;i++)                                     //页面加载完成后展开卡片
09     {
10         var tdeg=(i<=n/2)?-(n/2-i)*deg:(i-n/2)*deg;          //分为两部分计算度数
11         cards[i].style.webkitTransform="rotate(" + tdeg+ "deg)"; //旋转度数
12         cards[i].style.msTransform="rotate(" + tdeg+ "deg)";
13         cards[i].style.MozTransform="rotate(" +tdeg + "deg)";
14         cards[i].style.OTransform="rotate(" + tdeg+ "deg)";
15         cards[i].style.transform="rotate(" + tdeg + "deg)";
16     }
17     for (var i=0;i<n-1;i++)
18     {
19         cards[i].onclick=function (){                        //卡片点击事件
20             change(index(this,cards));                        //更改卡片位置函数
21         };
22     }
23     cards[n-1].onclick=function (){
24         click++;                                              //点击次数增加
25         for (var i=0;i<n;i++)
26         {
27             if (click%2)                                       //判断点击奇数次与偶数次
28             {
29                 var tdeg=(i<=n/2)?-(n/2-i)*deg:(i-n/2)*deg;
30             }else{
31                 var tdeg=0;
32             }
33             cards[i].style.webkitTransform="rotate(" + tdeg+ "deg)";
34             cards[i].style.msTransform="rotate(" + tdeg+ "deg)";
35             cards[i].style.MozTransform="rotate(" +tdeg + "deg)";
36             cards[i].style.OTransform="rotate(" + tdeg+ "deg)";
37             cards[i].style.transform="rotate(" + tdeg + "deg)";
38         }

```

```

39  }
40  function change(index) {                                //改变位置函数
41  var tdeg;
42  for (var i=0;i<=index ;i++)                             //该卡片与之前的卡片
43  {
44  tdeg=-(index-i)*deg;
45  cards[i].style.webkitTransform="rotate(" + tdeg+ "deg)";
46  cards[i].style.msTransform="rotate(" + tdeg+ "deg)";
47  cards[i].style.MozTransform="rotate(" +tdeg + "deg)";
48  cards[i].style.OTransform="rotate(" + tdeg+ "deg)";
49  cards[i].style.transform="rotate(" + tdeg + "deg)";
50  }
51  for(var i=index+1;i<n;i++)
52  {                                                         //该卡片之后的卡片多旋转几度，便于该卡片文字的展示
53  tdeg=(i-index)*deg+5;
54  cards[i].style.webkitTransform="rotate(" + tdeg+ "deg)";
55  cards[i].style.msTransform="rotate(" + tdeg+ "deg)";
56  cards[i].style.MozTransform="rotate(" +tdeg + "deg)";
57  cards[i].style.OTransform="rotate(" + tdeg+ "deg)";
58  cards[i].style.transform="rotate(" + tdeg + "deg");}}}

```

5.15 回到页面的顶部

回顶部效果使用户能够从较长网页的任何位置快速回到页面的最顶端，一般使用一个向上的标志提示。这一标志是一个绝对定位的 div，它总能固定在窗口的右下角。

```

//CSS 代码
01  #bottom-top{
02  position:absolute;right:0;
03  width:34px;height:34px;
04  background:url("img/bottom-top.png") no-repeat;
05  background-position:top;
06  }

```

在页面加载完成后，通过 JavaScript 获取到当前窗口的高度，计算 div 的 top 值并应用在 CSS 上，当页面滚动时立即获得滚动条离开上部的距离，将此数值附加在 div 的 top 值上即可使得 div 总是处在右下角。另外使用 JavaScript 鼠标悬浮事件改变鼠标指向 div 时的背景。主要的 JavaScript 代码为：

```

01  var bottom_top=document.getElementById('bottom-top');    //获取 div
02  var bottom_top_height=bottom_top.offsetHeight;           //获取 div 本身高度
03  var window_height=win('height');                         //获取可视区高度
04  bottom_top.style.top=window_height-bottom_top_height+"px"; //初始化 div 的 top 值
05  bottom_top.onmouseover=function(){                        //鼠标悬浮

```



```

06  this.style.backgroundPosition='bottom';
07  }
08  bottom_top.onmouseout=function(){           //鼠标离开
09  this.style.backgroundPosition='top';
10  }
11  bottom_top.onclick=function(){             //点击
12  document.documentElement.scrollTop=0;       //回到顶部
13  document.body.scrollTop=0;
14  }
15  window.onscroll=function(){                //页面滚动
16  var scroll_dis=win('scrollTop');           //获取滚动距离
17  var top=window_height-bottom_top_height+scroll_dis;
18  bottom_top.style.top=top+"px";            //重置 top 值
19  }

```

注意：有关各种浏览器下获取可视区大小等 JavaScript 内容请自行查找相关知识或者查看实例源代码。

5.16 拖曳和抛出

本节讲述网页元素的拖曳和抛出的实现，并附带网页内小窗口的实现，最后将两者结合起来制作一个可以随意拖曳和抛出的通知窗口，抛出后该窗口根据鼠标拖曳的方向和速度继续运动，碰到屏幕反向运动，最终停靠在屏幕的下方。窗口的样式如图 5.32 所示。

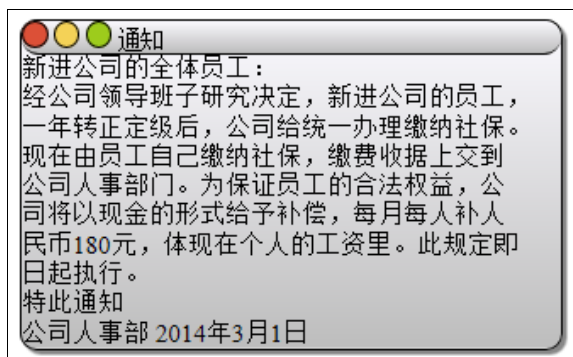


图 5.32 窗口实现

5.16.1 拖曳实现原理

当 HTML 元素采用绝对定位时，该元素就会脱离文档流，使用定位值即可到达各种位置。在 JavaScript 的 onmousedown 事件被触发时，通过 event 事件获取到鼠标的位置，当鼠标拖动时，获取鼠标移动的距离计算出定位值应用在元素上。

5.16.2 抛出与模拟抛物原理

元素运动中有一个速度的概念，有了速度运动实现就简单了。在鼠标拖曳时，相邻两点的距离是不固定的，从而可以通过相邻两点的位置做计算转换为速度。在鼠标按钮松开后，根据速度值使元素运动，元素的运动过程中需判断元素是否碰到了屏幕边缘，当元素到达边缘时速度反向并减小。多次碰撞后，速度便会为 0，这也就是碰撞运动。

注意：限于篇幅和侧重点的原因，JavaScript 部分的实现见实例代码，大致的思路实现在之前的很多 JavaScript 代码中都有所体现。

5.16.3 窗口实现

一个窗口的必须元素有包括最小化、最大化及关闭按钮的标题栏和窗口的文本区。HTML 部分可以使用多个 div 组合起来。为了使窗口能够应用在多个场合，元素使用 class，使用时将整个窗口实现代码放入一个 div 中即可。

```
01 <div class="window">
02 <div class="window_top">
03 <span class="window_top_close" title="关闭"></span>
04 <span class="window_top_min" title="最小化"></span>
05 <span class="window_top_max" title="合适大小"></span>
06 窗口标题
07 </div>
08 <div class="window_main">窗口文本区</div>
09 </div>
```

对窗口进行样式与布局的设定如下，其中包括第 1~7 行的整体设置、第 8~13 行的标题栏整体和第 14~22 行的按钮设置。第 15 行使得所有的按钮并排在一行中。实例的代码中只使用了背景颜色的不同区分按钮，也可使用不同的背景图片或者伪对象设置不同的按钮样式。

```
01 .window{                                /*窗口*/
02     position:relative;
03     border:1px solid;border-radius:10px;
04     background:#E2E2E2;                  /*背景与渐变背景*/
05     background:linear-gradient(to bottom ,#FBFAFF, #BDBDBD);
06     box-shadow: 2px 2px 2px #888888;
07 }
08 .window_top{                             /*标题区域*/
09     width:100%;
10     border-radius:10px;border-bottom:1px solid;
11     font-size:0.9em;
12     background:linear-gradient(to bottom ,#FBFAFF, #BDBDBD);
13 }
```

```
14 .window_top_close,.window_top_min,.window_top_max{ /*最大化、最小化、关闭按钮*/
15     display:inline-block;                          /*按钮同排显示*/
16     height:1em;width:1em;
17     border:1px solid;border-radius:10px;
18     background:#FEFFFD;
19 }
20 .window_top_close{background:#DC5037;}
21 .window_top_min{background:#F4D257;}
22 .window_top_max{background:#9DCC1C;}
```

第 6 章 页面的布局

设计一个新的页面，首先要考虑的就是布局的合理性。页面布局主要包括设计首页、栏目页、内容页、页脚、滚动条、表格等页面主要元素的位置。合理的页面布局能给网站带来一定的感官优势，有句话说得好，一个好的页面能让你 1 秒内留住用户，而一个不好的页面能在 3 秒钟内毁掉一个用户。

本章主要涉及的知识点有：

- 图文混排
- 几种不同的居中布局方法
- 由浮动引起的布局问题的解决方案
- 绝对定位与相对定位
- 适配 iPad 屏幕的布局
- 经典的 Clearfix 和升级版的 Clearfix
- 滚动条的控制
- CSS 3 文本分列
- Flash 参数设定引起的布局问题
- Metro 和 Flexbox 布局风格

6.1 图文混排

文字与图片混排是较为常见的一种网页布局方式，其中，图片能够给予用户现场感，使得所呈现的内容更加生动，而文字则对于图片未能充分展示的细节进行解释说明，使得页面主旨内容更加充实。因此，图文混排的方式是很多新闻类型页面主要采用的页面布局。

图文混排实现的是将文字与图片混合排列，图片可以在文字的一侧，也可以嵌入在文字内部。在 Word 中，可以用环绕方式来让文字显示在图片周围，在 CSS 中，则可以用 float 来让文字在没有清理浮动时，显示在图片以外的空白处。下面以左图右文为例来进一步说明。左图右文的呈现效果有两种，分别如图 6.1、图 6.2 所示。

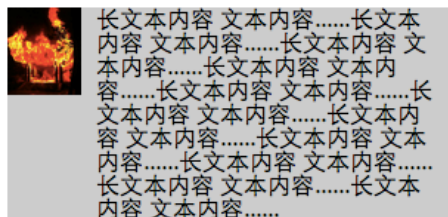


图 6.1 图文混排展示效果一

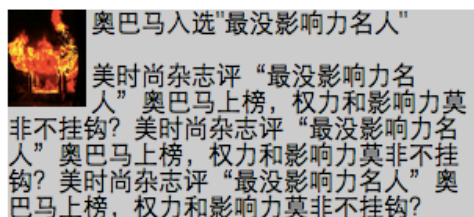


图 6.2 图文混排展示效果二

图 6.1 效果的实现代码如下：

```
//HTML 代码
01 <div id="wrap">
02   <div id="picture"></div>
03   <div id="content">长文本内容.....</div>
04 </div>
//CSS 代码
05 #wrap{
06   background:#ccc;      /* 阴影效果*/
07   width:300px;
08   clear:both;           /* 清除浮动*/
09 }
10 #picture{
11   float:left;           /* 图片向左浮动*/
12   width:50px;           /* 设置图片包裹器的宽度*/
13   height:50px;         /* 设置图片包裹器的高度*/
14 }
15 #content{
16   float:left;           /* 文字向右浮动*/
17   width:230px;          /* 设置文字包裹器的宽度*/
18   margin-left:10px;     /* 设置文字包裹器的宽度*/
19 }
```

在代码第 16 行中，图片与文字的包裹器同时将 float 设置成 left，图片节点定义则在前显示在左侧，文字节点定义在后则显示在右侧（所以要注意 CSS 代码的先后顺序）。但是，从图 6.1 可以看出，文字节点为块元素，文字内容折行后不会填充到图片下方的空间。如要呈现类似于“顶格”显示的效果，需消除图片下方的空白部分，请继续往下看。

图 6.2 效果的实现方法一，代码如下：

```
//HTML 代码
01 <div id="wrap">
02   <div id="picture"></div>
03   <div id="content">长文本内容.....</div>
04 </div>
//CSS 代码
05 #wrap{
06   background:#ccc;      /* 阴影效果*/
```

```

07     width:300px;
08     margin-bottom:100px;
09     clear:both;
10 }
11 #picture{
12     float:left;                /*图片向左浮动*/
13     width:50px;                /*设置图片包裹器的宽度*/
14     height:50px;              /*设置图片包裹器的高度*/
15 }

```

在代码第 12 行中，仅设置图片节点向左浮动，而不设置文字节点的浮动样式，则文字内容即可占用图片下方的空间，呈现“顶格”效果。

图 6.2 效果的实现方法二，代码如下：

```

//HTML 代码
01 <div id="wrap2">
02     <a href="#" target="_blank"></a>
03     <span>奥巴马入选"最没影响力名人"</span>
04     <p>美时尚杂志评“最没影响力名人” .....</p>
05 </div>
//CSS 代码
06 #wrap2{
07     width:300px;background:#ccc;
08 }

```

这种实现方案的重点在于 HTML 的结构代码，其强调使用行内元素。行内元素包括 `<a>`、`` 等，此类元素又被称为内联元素，默认呈现在行内，如代码第 2~3 行的 `<a>` 和 `` 同时在一行内显示，从而得以呈现出左图右文的效果。

6.2 文本内容垂直居中

第 1 章我们曾经讲过文字的对齐方式，这里再来深入了解一下。许多网页设计中都会要求文字内容垂直居中，例如在网页的菜单中文字元素必须在垂直位置上居中。未居中的文本如图 6.3 所示，给人感觉是文字居上显示。居中后的效果如图 6.4 所示，可以看出文字在垂直位置上居中呈现的效果。

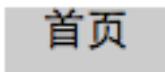


图 6.3 文字内容未居中

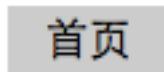


图 6.4 文字内容居中

传统的居中方案一般可以通过调节 `padding-top`、`padding-bottom`、`margin-top`、`margin-bottom` 属性值，对文字内容距离上方和下方的间距进行设置，从而实现垂直方向上居中的呈现。这种方法的缺点是，当浏览器的设置发生变化时，这种居中呈现的效果并不是绝对的，且在实现代码时对于间距属性的具体值难以把握，提高了代码难度。

为了实现真正意义上的居中，需同时设置文字内容元素的 `height` 与 `line-height` 属性，并让二者的属性值相等。文字居中的实现代码如下：

```
//HTML 代码
01 <div id="menu">首页</div>
//CSS 代码
02 #menu{
03     background:#ccc;
04     height:25px;
05     line-height:25px;
06     width:60px;
07     text-align:center;
08 }
```

从代码段中可以看出，代码第 4、5 行分别设置文字内容元素的 `height` 属性和 `line-height` 属性，文字内容便会自动居中。此处唯一要注意的就是，`height` 与 `line-height` 的数值必须相等，本例都是 25px。

6.3 自适应宽度的水平居中

宽度自适应的水平居中在网页制作中非常常见，而且是较为实用的一种设计方法。宽度自适应就是元素的宽度可以根据元素包含的内容进行自动调整。下面就介绍两种实现自适应宽度的水平居中。

(1) 当父元素和子元素都没有定义宽度的情况下实现水平居中：

```
//HTML 代码
01 <div class="navbar">
02 <ul>
03     <li><a href="#">首页</a></li>
04     <li><a href="#">资讯</a></li>
05     <li><a href="#">人物</a></li>
06     <li><a href="#">综合</a></li>
07 </ul>
08 </div>
//CSS 代码
09 .navbar {
10     text-align:center;    /*设定居中*/
11 }
12 .navbar ul {
13     display:inline-block; /*将对象呈递为内联对象，但是对象的内容作为块对象呈递*/
14     *display:inline;     /*支持 IE8 以下版本浏览器*/
15     *zoom:1;             /*支持 IE8 以下版本浏览器*/
16 }
17 .navbar li {
```

```

18     float:left;
19 }
20 .navbar li + li {
21     margin-left:20px;
22 }

```

从 CSS 代码可以看出，在同时都未设定父元素和子元素的宽度的情况下，即父子元素都自适应宽度的情况下，要使得子元素居中，可以使用 `text-align:center` 和 `display:inline-block` 相结合的方式。在父元素上设定 `text-align:center`，在子元素上设定 `display:inline-block`。其中，`display:inline-block` 在 IE 系列中只有 IE 8 及其以上版本支持，要使得 IE 6、IE 7 支持，需要在对应的元素样式列表中加入 `*display:inline; *zoom:1`，以使得在 IE 6、IE 7 下，`display:inline` 呈现的效果与 `display:inline-block` 的效果一致。

（2）另一种自适应方式利用 `display:table` 来实现，可以使用较少的标签：

```

//HTML 代码
01 <div class="navbar">
02 <ul>
03     <li><a href="#">首页</a></li>
04     <li><a href="#">资讯</a></li>
05     <li><a href="#">人物</a></li>
06     <li><a href="#">综合</a></li>
07 </ul>
08 </div>
//CSS 代码
09 .navbar {
10     display:table;
11     margin:0 auto;
12 }
13 .navbar li {
14     display:table-cell;
15 }
16 .navbar li + li {
17     padding-left:20px;
18 }

```

在代码段第 10 行定义了 `display` 属性值为 `table`，读者可以通过运行代码段来对比两种居中方法的差异。

注意：`display:table` 不支持 IE 7 及以下版本的浏览器，其他的主流浏览器都支持。

6.4 固定宽度且居中

对固定宽度的元素设定居中，通常应用在整个网页的居中布局中。一般来说，将整个页面内容使用一个元素包裹起来，设定宽度，再进行居中，这是传统网站常用的设计实现

方法。当然，在页面的局部固定宽度的内容中，如需居中，也仍然可以使用该方法。实现代码如下：

```
//HTML 代码
01 <div class="content">主要内容.....</div>
//CSS 代码
02 .content{
03     margin:0 auto;      /*设定元素左、右 margin 均为 auto*/
04     width:1004px;      /*固定宽度*/
05 }
```

其中，class 为 content 的元素即为包裹页面内容的容器，固定宽度为 1004px，第 3 行代码同时设定元素左、右 margin 必须均为 auto，上、下 margin 可为任意值，从而达到居中的效果。

6.5 固定页脚

固定页脚的设计对于全站各个级别（包括首页、内容页、详情页等）的页脚引入有很重要的帮助。页脚只需要设计和制作一次，就能应用到网站的所有页面，给全站呈现出一个完美的结尾和统一的效果。



图 6.5 固定页脚

HTML 代码如下：

```
//HTML 代码
01 <div class="container">
02     <div id="header">Header Title</div>
03     <div id="page" class="clearfix">
04         <div id="left">Left Sider</div>
05         <div id="content">Main Content</div>
06         <div id="right">Right Sider</div>
```

```
07     <div id="footer">Footer Section</div>
```

```
08 </div>
```

CSS 代码如下:

//CSS 代码

```
01  html,body {                                /* reset: 去除 html, body 的 margin, padding */
02      margin: 0;
03      padding:0;
04      height: 100%;                          /*设置 height 为 100%, 便于后续子元素百分比的设置*/
05  }
06  .container {
07      min-height:100%;
08      height: auto !important;
09      height: 100%;                          /*IE 6 不识别 min-height*/
10      position: relative;                    /*便于该 container 包含的元素进行绝对定位*/
11  }
12  #header {
13      background: #ff0;
14      padding: 10px;
15  }
16  #page {
17      width: 960px;
18      margin: 0 auto;
19      padding-bottom: 60px;                  /*等于 footer 的高度*/
20  }
21  #footer {
22      position: absolute;                    /*页脚相对于 container 绝对定位*/
23      bottom: 0;                            /*让页脚固定在容器 container 的底部*/
24      width: 100%;                          /*设定页脚宽度自适应*/
25      height: 60px;                         /*页脚的高度固定*/
26      background: #6cf;
27      clear:both;
28  }
29  /*=====主体内容部分=====*/
30  #left {
31      width: 220px;
32      float: left;
33      margin-right: 20px;
34      background: lime;
35  }
36  #content {
37      background: orange;
38      float: left;
39      width: 480px;
40      margin-right: 20px;
```

```

41 }
42 #right{
43     background: green;
44     float: right;
45     width: 220px;
46 }

```

此处固定页脚的实现原理是，页面的 header、page、footer 均包含于 container 容器内部，由于 container 容器设置 position:relative，因而再设置 footer 为 position:absolute 即可使得 footer 相对于 container 容器固定定位，进一步设置 footer 的 bottom:0 即可使得 footer 固定于 container 容器的底部。

注意：id 值为 page 的 div 元素有一个较为关键的设置，在第 19 行可见，在该容器上设置一个 padding-bottom 值，而且这个值要等于（或略大于）页脚 footer 的 height 值，且不能使用 margin-bottom 来代替 padding-bottom，不然会无法实现效果。

6.6 控制位置：绝对位置和相对位置

CSS 中有 3 种基本的定位机制：普通定位、浮动和绝对定位。默认情况下，所有元素都是普通定位。相对定位是普通定位模型中的一种，并且相对定位的元素是相对于它在普通定位中的位置。如果一个元素进行相对定位，它将占据原有位置，随后可以通过设置垂直或水平位置，使得该元素相对于原有位置进行移动。

```

//CSS 代码
01 #wrap{
02     position:relative;
03     top:10px;
04     left:20px;
05}

```

以上代码定义的元素是相对定位，将其 top 设置为 10px，那么该元素将呈现在距离原有位置顶部下方 10px 的位置。将其 left 设置为 20px，那么该元素将呈现在距离原有位置左侧 20px 的位置。效果如图 6-6 所示。

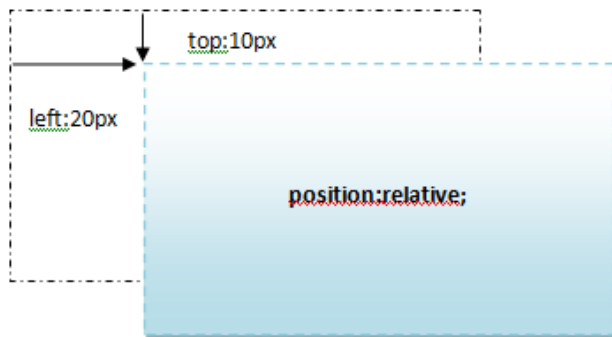


图 6.6 相对定位元素

与相对定位不同的是，绝对定位的元素的位置与文档流无关，且不占据空间。绝对定位元素的位置是相对于距离它最近的已有定位的父元素。如果该元素没有已定位的父元素，那么它将相对于 HTML 元素定位。与相对定位元素的相同之处是，绝对定位元素也是通过设定 top、right、bottom、left 来控制距离相对点的上、右、下、左的距离。

```
//CSS 代码
01  #logo{
02      width:500px;
03      height:100px;
04      position:relative;    /*对父元素设定相对定位*/
05  }
06  #logo .photo{
07      position:absolute;    /*对子元素设定绝对定位*/
08      right:20px;           /*设定子元素距离父元素右侧的值*/
09      bottom:10px;          /*设定子元素距离父元素底部的值*/
10  }
//HTML 代码
11  <div id="logo">
12      <div class="photo">This is the first logo.</div>
13  </div>
```

注意：由于绝对定位元素不占据文档流的空间，从而不会影响到普通定位的元素的位置，但是可能会覆盖到页面上的其他元素。可以通过设置 z-index 属性来控制元素的叠放次序。

6.7 一个图文混排的网页选项卡

第 3 章我们曾经介绍过 TAB 标签页，主要讲解了标签链接的形式。本节的 TAB 选项卡在网页中主要用于把同类并列的信息展示在一个局部位置内。利用选项卡，可以为每一项显示较为详细的内容。下面介绍一种使用 li 列表、可以图文混排的选项卡解决方案，可以使得图片与文字交叉显示，提高 TAB 的实用性与生动性。

选项卡的展示效果如图 6.7 所示，图中定义了 5 个选项卡，突出显示当前选项卡的特征效果，每个选项卡的详细内容部分都含有图片与文字的内容，并且图片与文字不是各自占一行，而是进行混合排列，以提高图片与文字的关联性，便于用户阅读。

HTML 代码如下：

```
//HTML 部分代码
01  <div id="gallery">
02      <div class="on" title="news"><span>新闻</span></div>
03      <div class="off" title="entertainment"><span>娱乐</span></div>
04      <div class="off" title="travel"><span>旅游</span></div>
05      <div class="off" title="city"><span>城市</span></div>
06      <div class="off" title="life"><span>生活</span></div>
```

```

07 </div>
08 <div id="news" class="show">
09     <p>Although he showed...</p>
10     
11 </div>
12 <div id="entertainment" class="hide">
13     <p>His youth was ...</p>
14     
15 </div>
16 <div id="travel" class="hide">
17     <p>Gogh, Vincent...</p>
18     
19     <p>...</p>
20 </div>
21 <div id="city" class="hide">
22     
23     <p>Russian-born French ...</p>
24 </div>
25 <div id="life" class="hide">
26     
27     <p>....</p>
28 </div>

```

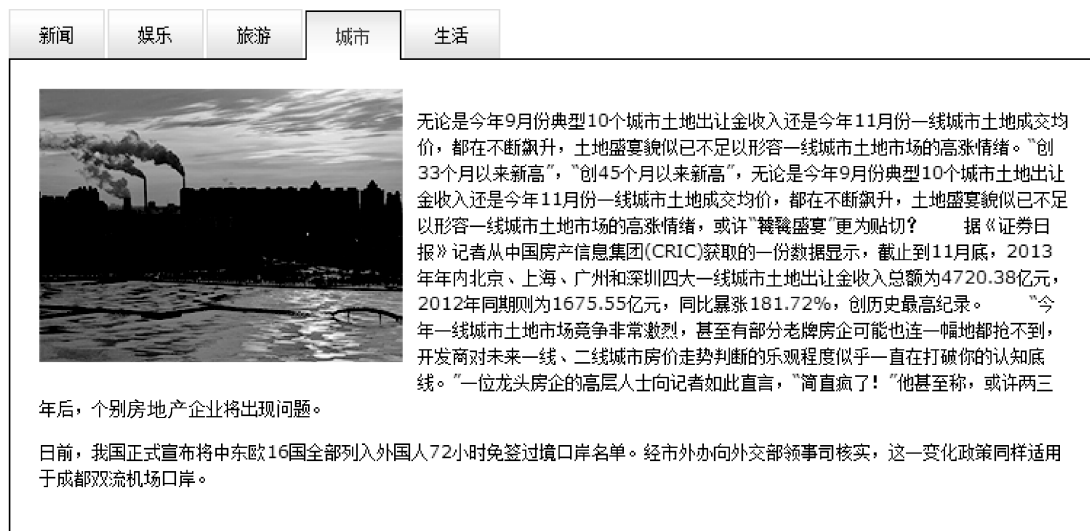


图 6.7 图文混排选项卡

CSS 代码如下：

```

01 .clear {
02     clear: both;
03 }
04 #gallery {

```

```

05     font: 11px verdana,arial,sans-serif;           /*设置字体大小及字体类型*/
06     width: 750px;
07     padding: 15px 0 0 0;
08     line-height: 15px;                             /*设置行高*/
09 }
10 #gallery .on { /*当前选项卡的样式表*/
11     color: #c00;
12     padding: 0 20px;
13     margin-right: 2px;
14     margin-top: 1px;
15     float: left;
16     background: url(tabs_2.gif) repeat-x left bottom; /*当前选项卡的背景图*/
17     border: 1px solid #000;                          /*设置边框*/
18     border-bottom: 0;                                /*不显示底部边框*/
19     cursor: pointer;                                 /*手势*/
20     height: 33px;
21     line-height: 32px;
22     position: relative;
23     z-index: 100;                                   /*叠放次序*/
24 }
25 #gallery .off {                                     /*非当前选项卡的样式表*/
26     color: #000;
27     height: 33px;
28     margin-right: 2px;
29     line-height: 33px;
30     padding: 0 20px;
31     float: left;
32     background: url(tabs_0.gif) repeat-x left bottom; /*非当前选项卡的背景图*/
33     border: 1px solid #ddd;
34     border-bottom-color: #000;
35     cursor: pointer;
36     position: relative;
37     z-index: 20;
38 }
39 .show {                                             /*显示选项卡内容的样式表*/
40     clear: left;
41     background: #fff;
42     width: 708px;
43     margin-top: 0;
44     top: -1px;
45     border: 1px solid #000;
46     padding: 20px;
47     position: relative;

```

```

48     z-index: 50;
49     font: 11px verdana, arial, sans-serif;
50     line-height: 18px;
51 }
52 .show img {                                /*图文混排中图片的样式*/
53     float: left;
54     margin: 0 10px 10px 0;
55 }
56 .hide {                                    /*不显示选项卡内容的样式表*/
57     display: none;
58     width: 0;
59     overflow: hidden;
60 }

```

JavaScript 代码如下:

```

01  onload = function() {
02      var e, i = 0;
03      while (e = document.getElementById('gallery').getElementsByTagName ("DIV") [i++]) {
04          if (e.className == 'on' || e.className == 'off') {
05              e.onclick = function () {
06                  var getEls = document.getElementsByTagName("DIV");
07                  for (var z=0; z<getEls.length; z++) {
08                      getEls[z].className=getEls[z].className.replace('show', 'hide');
09                      getEls[z].className=getEls[z].className.replace('on', 'off');
10                  }
11                  this.className = 'on';
12                  var target = this.getAttribute('title');
13                  document.getElementById(target).className = "show";
14              }
15          }
16      }
17  }

```

在选项卡的制作过程中, 一个好的 HTML 结构有助于将选项卡实现逻辑简单化。在本实例中, 使用元素 `gallery` 包裹所有的选项卡标题, 并且选项卡标题中自定义 `title` 属性与对应的选项卡内容的 `id` 属性一致。选项卡标题使用样式表 `on` 与 `off` 分别表示当前选项卡与非当前选项卡, 选项卡内容用样式表 `show` 与 `hide` 分别表示显示选项卡内容与隐藏选项卡内容的样式。在选项卡内容中, 用元素 `div` 包裹元素 `p` 与 `img` 标签, 分别用于图文内容的排列, 并且设置图片向左浮动 `float:left`, 此布局的详细介绍见 6.1 介绍的图文混排实例。

在 CSS 代码中, 元素 `gallery` 元素中定义的多种字体类型, 会按照客户端所支持的字体依次显示, 具体来说, 由于预先定义 `verdana` 字体, 如果用户浏览器不支持 `verdana` 字体, 则会按照 `arial` 字体显示, 如果仍然不支持 `arial` 字体, 则会用 `sans-serif` 字体显示。在选项卡上设置叠放次序的属性 `z-index`, 只有在相对定位或绝对定位的元素上才能生效, 因此,

选项卡均设置属性 `position:relative`。

6.8 兼容浏览器的最小高度

在网页设计的过程中，往往会遇到需要设置最小高度的问题。例如，设计方案中某个 `div` 是有背景图的，如果 `div` 高度太小，背景图就无法完整显示，这会影响视觉效果。为了设置这种类型的 `div` 的最小高度，即当该 `div` 内容小于最小高度时，该 `div` 显示的高度等于最小高度设定的值，当 `div` 内容超过最小高度时，该 `div` 高度会自动根据内容进行扩展。最小高度需要用到 CSS 的 `min-height` 属性。

`min-height` 属性的使用说明是设置或检索对象的最小高度。如果 `min-height` 的值大于 `max-height` 的值，高度会被自动设定为 `max-height` 的值。

最简单的最小高度解决方案，采用以下 CSS 样式：

```
//CSS 代码
01 min-height {
02   min-height:100px;
03 }
//HTML 代码
04 <div class="min-height">
05   <pre>
06     line
07     line
08     .....
09   </pre>
10 </div>
```

本来单用 `min-height` 就可以了，但是 IE 6 不支持 `min-height` 属性，因此需要加两行代码 `height:auto !important;` 和 `height:100px;`，对 IE 6 下的最小高度进行修正，其中 `!important` 的作用是对除了 IE 以外的浏览器来说的，任何后面标有 `!important` 的语句将获得绝对的优先权。改进后的代码在 IE 6/IE 7/IE 8/Firefox/Opera/Chrome/Safari 下测试有效，如下：

```
//CSS 代码
01 .min-height {
02   min-height:100px;
03   height:auto !important; /*支持 IE 6*/
04   height:100px;          /*支持 IE 6*/
05 }
//HTML 代码
06 <div class="min-height">
07   <pre>
08     line
09     line
```



```

10      .....
11      </pre>
12 </div>

```

在 CSS 代码第 3、4 行针对 IE 6 进行特殊设置。

6.9 让 div 显示在屏幕的中央

在网页制作的过程中，还有一种较为常见的应用是，在不知道屏幕宽度的情况下，希望把一些较为重要的元素放置在屏幕中央。因此该问题可以简述为如何把 div 放到屏幕中间。具体的实现方法有很多种，这里举其中的一种实现方法作为范例：

```

//CSS 代码
01 .popup {
02     width: 500px;
03     height: 400px;
04     position: absolute;
05     top: 50%;
06     left: 50%;
07     margin-top: -200px;    /*设置为 height 值的一半*/
08     margin-left: -250px;   /*设置为 width 值的一半*/
09     border: 1px solid #fffcc;
10 }
//HTML 代码
11 <div class="popup">Something so important!</div>

```

以上代码段实现了 div 在屏幕中央显示的效果。可以看出，在样式表中制定了元素 div 的宽度、高度，且设置为绝对定位，top、left 值均为 50%。

注意：必须明确指定 div 的宽度和高度，并且设置 margin-left 和 margin-top 属性分别为宽度和高度值的一半，这样就可以使得该 div 呈现到屏幕的中央。

增加一项要求，当 div 高度大于显示器高度的时候，将其显示在顶端，div 高度是随内容多少变化的。

CSS 代码如下：

```

01 html,body {
02     width: 100%;
03     height: 100%;
04     padding: 0;
05     margin: 0;
06     overflow: auto;
07 }
08 .wrap{
09     height: 100%;
10     text-align: center;

```

```

11 }
12 .container {
13     width:200px;
14     border:1px solid #ccc;
15     text-align: center;
16     display: inline-block;
17     vertical-align: middle;
18 }
19 .edge {
20     width:1px;
21     *width:0;
22     height:100%;
23     display: inline-block;
24     vertical-align: middle;
25 }

```

HTML 代码如下:

```

01 <body>
02     <div class="wrap">
03         <span class="edge"></span>
04         <span class="container">
05             <br/>超实用的 CSS 代码段<br/>
06         </span>
07     </div>
08 </body>

```

在 HTML 结构中,使用两个并列的 display-inline 元素,即.edge 与.container 两个元素,display-inline 的好处在于既具有行内元素在一行内显示的特点,又具有块元素的宽、高属性。其中,父元素.wrap 的高度设置为 100%。元素.edge 的高度设置为 100%,撑满屏幕,并且设置 vertical-align 为 middle,即中线与父容器中线对齐。类似地,.container 元素 vertical-align 为 middle,即中线与父容器中线对齐,即呈现为居中显示。具体实现效果如图 6.8 所示。



图 6.8 元素放置在屏幕中央

6.10 iPad 屏幕布局

通过移动设备访问互联网的用户量正超过 PC 设备，并且移动终端的多样化导致全球用户访问互联网的设备也越来越多样化。据有关市场研究公司的调查结果显示，使用平板电脑访问互联网的用户数持续增长，这让网页制作人员对平板电脑进行适配提出了新的要求。以下就以 iPad 屏幕的布局为例，对移动设备网页制作的布局进行说明。

首先，将网页宽度设置为默认等于屏幕宽度：

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

viewport 是网页默认的宽度和高度，上面这行代码中 width=device-width 的意思是网页宽度默认等于屏幕宽度，原始缩放比例（initial-scale=1）为 1.0，即网页初始大小占屏幕面积的 100%。

然后，利用 CSS 3 引入的 Media Query 模块，即根据自动探测屏幕宽度的原理进行选择加载 CSS。

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link rel="stylesheet" type="text/css" media="screen and (max-width: 479px)"
href="index-banner320.css" />
<link rel="stylesheet" type="text/css" media="screen and (min-width: 479px) and (max-width:
639px)" href="index-banner480.css" />
<link rel="stylesheet" type="text/css" media="screen and (min-width: 639px)"
href="index-banner640.css" />
```

以上代码会根据屏幕的不同宽度加载不同的 CSS 文件，分别为小于 480px、大于等于 480px 且小于 640px、大于等于 640px 的 3 种屏幕宽度加载各自的 CSS 文件。

在 CSS 文件中，也可以根据不同的屏幕分辨率，选择应用不同的 CSS 规则。

```
//CSS 代码
01 @media only screen and (device-width: 768px) {
02     /*适合于普遍 iPad 布局*/
03 }
04 @media only screen and (min-device-width: 481px) and (max-device-width: 1024px) and
    (orientation:portrait) {
05     /*仅适合于 iPad 竖屏布局*/
06 }
07 @media only screen and (min-device-width: 481px) and (max-device-width: 1024px) and
    (orientation:landscape) {
08     /*仅适合于 iPad 横屏布局*/
09 }
```

iPad 上的 Webkit 内核支持基于方向（orientation）的 CSS 媒体查询（media query）声明，因此，可以分别对 iPad 的竖屏、横屏布局构造不同的样式表，以便展示不同效果。以上代码的原理即是根据横屏与竖屏的屏幕宽度不同（竖屏 orientation 属性为 portrait、横屏时 orientation 属性为 landscape）进行屏幕的适配，以匹配不同的样式表，从而进一步控制横屏、竖屏的网页内容的呈现。图 6.9 与图 6.10 分别展示了同一个页面在 iPad 中横屏与竖

屏的显示效果。



图 6.9 iPad 横屏展示效果



图 6.10 iPad 竖屏展示效果

6.11 经典的 CSS Clearfix

当元素设置 `float` 属性时，该元素所在的物理位置已经脱离文档流，从而影响页面布局。为了让文档流能识别 `float` 元素，且 `float` 元素后面的元素不被浮动所影响，可以使用 `clear:both` 来清除浮动，为清除浮动单独定义一个 CSS 样式，即 `clearfix`：

```
//CSS 代码
01 .clearfix{
02     clear: both;
03 }
```

然后使用 `<div class="clearfix"></div>` 来专门清除浮动。但是，由于单独添加的 `<div class="clearfix"></div>` 标签，在 IE 和 Firefox 下会引起高度的变化，需要通过如下方法解决：

```
//CSS 代码
01 .clear fix{
02     clear: both;
03     height:1px;
04     margin-top:-1px;
05     overflow:hidden;
```

```
06 }
```

另外，`overflow: hidden` 也可以用于清除浮动，但是 `overflow` 会额外带来一些副作用，因此不适合广泛使用。

当父级元素使用 `overflow: hidden` 时，如果其子元素定位到部分或全部在父元素之外，父元素就会对超出其外的子元素部分进行裁剪。对 CSS 3 来说，`overflow: hidden` 也会对一些属性产生影响，例如当父元素使用 `overflow: hidden` 属性时，`box-shadow` 会被裁剪。

6.12 升级版的 Clearfix

由于经典的 CSS Clearfix 需要单独定义一个 `<div class="clearfix"></div>` 标签来专门清除浮动，影响了 HTML 语义化的结构安排。为此，专业人士提出了升级版 Clearfix 的构想。具体代码如下：

```
//CSS 代码
01 .clearfix:after {
02     content: ".";           // 内容为空
03     display: block;
04     height: 0;
05     clear: both;
06     visibility: hidden
07 }
08 .clearfix {
09     display: inline-table;
10 }
11 /* Hides from IE-mac */
12 //为 Mac 上的 IE 做兼容
13 * html .clearfix {
14     height: 1%;
15 }
16 .clearfix {
17     display: block
18 }
19 /* End hide from IE-mac */
```

升级版的 Clearfix 不需要创建额外的专门用于清除浮动的节点，只需要在需要清除浮动的父容器上添加名为 `clearfix` 的 class 即可，这适用于大多数符合标准的浏览器。升级版的 Clearfix 的主要原理是使用伪元素 `:after` 创建一个隐形的块元素来清除浮动，且设置该隐形的块元素 `content: "."`，即设置内容为空。

第 8~10 行的样式为 `clearfix` 应用 `inline-table` 显示属性，它不兼容 IE 6 与 IE 7 浏览器。代码第 13~18 行通过符号 `*`，使得这段兼容性代码在 Mac 上的 IE 浏览器中是隐藏的。由于 IE 6、7 都不支持 `:after` 伪类，因此，通过第 14 行与第 17 行代码来触发 IE 6、7 的 `hasLayout`，

以清除浮动，但是 IE 8 支持:after 伪类，因此只需要针对 IE 6、7 进行修复。hasLayout 是 IE 的特有属性，即拥有布局样式。第 17 行对非 Mac 上的 IE 浏览器设置 display 为 block 属性，即设置为块元素。

由于目前 Mac 上的 IE 浏览器较少，因此去除对 Mac 上的 IE 浏览器支持之后，可以使用清除浮动方法，即通过使用 zoom 属性，具体代码如下：

```
//CSS 代码
01  *{
02      padding: 0px;
03      margin: 0px;
04  }
05  ul {
06      border: 3px solid #F00;
07  }
08  ul li {
09      width: 50px;
10      height: 50px;
11      float: left;
12      background-color: #00F;
13      list-style: none;
14      margin-right: 10px;
15  }
16  .clearfix{
17      zoom:1;
18  }
19  .clearfix:after{    // 采用伪类:after，对伪类层清除浮动
20      content:"";
21      display:block;
22      height:0;
23      line-height:0;
24      clear:both;
25      visibility:hidden;
26  }
//HTML 代码
27  <ul class="clearfix">
28      <li>xxx</li>
29      <li>xxx</li>
30      <li>xxx</li>
31      <li>xxx</li>
32      <li>xxx</li>
33  </ul>
```

这种 Clearfix 的原理是，在 IE 6、7 下 zoom: 1 会触发 hasLayout，从而使元素闭合内部的浮动。在标准浏览器下，.clearfix:after 这个伪类会在应用到.clearfix 的元素后面插入一个 clear: both 的块级元素，从而达到清除浮动的作用。在需要清除浮动时，只要写一个.clearfix

就行了，然后在需要清除浮动的元素中添加 `clearfix` 类名即可。

在 IE 6、7 下面只要是触发了 `hasLayout` 的元素就可以清除内部浮动。而在标准浏览器下面清除元素内部浮动的方法有很多，大多数的情况下使用 `clearfix:after` 都可以满足需求，所有主流浏览器都支持 `:after` 选择器。

6.13 强制垂直滚动条

利用 CSS 可以有效地显示或隐藏垂直及水平滚动条，下文具体介绍显示或隐藏垂直、水平滚动条的几种情况。

强制显示滚动条：

```
html {  
    overflow: scroll;  
}
```

强制隐藏滚动条：

```
html {  
    overflow: hidden;  
}
```

隐藏 IE 的水平滚动条：

```
html {  
    overflow-x: hidden;  
}
```

隐藏 IE 的垂直滚动条：

```
html {  
    overflow-y: hidden;  
}
```

强制显示 IE 的水平滚动条：

```
html {  
    overflow-x: scroll;  
}
```

强制显示 IE 的垂直滚动条：

```
html {  
    overflow-y: scroll;  
}
```

强制显示 Mozilla 的水平滚动条：

```
html {  
    overflow:-moz-scrollbar-horizontal;  
}
```

注意：仅仅强制显示水平滚动条，也就是说，即使需要显示垂直滚动条时，垂直滚动条也不会出现。

强制显示 Mozilla 的垂直滚动条：

```
html {
    overflow:-moz-scrollbars-vertical;
}
```

注意：仅仅强制显示垂直滚动条，也就是说，即使需要显示水平滚动条时，水平滚动条也不会出现。

6.14 CSS 3 文本分列

传统的解决多列布局的方法是拆分文本放在不同的列里边单独控制，这种方法非常的耗费时间。另一种解决方法就是通过 JavaScript 动态地判断文本并分割在不同列，这种方法普遍会出现不正常的情况。那么，如何能在不分割文本的情况下解决分列问题呢？CSS 3 columns 的参数为解决多列布局提出了新的思路。通过对 CSS 3 columns 参数的设置可以达到分列文字、定义分列线及列间距等效果。

目前 CSS 3 columns 这一属性的浏览器兼容情况如图 6.11 所示。

IE	Firefox	Chrome	Safari	Opera	IOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
7.0	16.0 -moz-	22.0 -webkit-	4.0 -webkit-	11.5	4.0-4.1 -webkit-		3.0 -webkit-		11.1		
8.0	17.0 -moz-	23.0 -webkit-	5.0 -webkit-	11.6	4.2-4.3 -webkit-		4.0 -webkit-		11.5		
9.0	18.0 -moz-	24.0 -webkit-	5.1 -webkit-	12.0	5.0-5.1 -webkit-		4.1 -webkit-		12.0		
10.0	19.0 -moz-	25.0 -webkit-	4.0 -webkit-	12.1	6.0 -webkit-	5.0-7.0	4.2 -webkit-	7.0 -webkit-	12.1	25.0 -webkit-	19.0 -webkit-
	20.0 -moz-	26.0 -webkit-	6.0 -webkit-					10.0 -webkit-			

图 6.11 CSS 3 文本分列浏览器支持

注意：column-rule 类似 border 属性。column-span:类似于 table 中的 rowspan。

图 6.11 展示了几种不同浏览器支持 CSS 3 columns 属性的情况，并且需要加前缀才可以识别，如基于 Webkit 内核的浏览器需在 columns 前面加上-webkit 前缀，基于 Mozilla 内核的浏览器需要加上-moz 前缀。

代码如下：

```
//CSS 代码
01 #columns-3 {
02     text-align: justify;
```



```
03      -moz-column-count: 3;
04      -moz-column-gap: 12px;
05      -moz-column-rule: 1px solid #c4c8cc;
06      -webkit-column-count: 3;
07      -webkit-column-gap: 12px;
08      -webkit-column-rule: 1px solid #c4c8cc
09  }
//HTML 代码
10  <div id="columns-3">目前，比特币价格指数(BPI)中的所有交易所比特币价格均低于 700 美元，
    而 BPI 本身自上周四以来已下跌近 20%，至 670 美元。其中，全球最大交易所“比特币中国”
    的价格上周五早间已跌至 4003 元人民币(约合 658 美元).....</div>
```

CSS 3 分列在网站上显示起来会非常不错，现实的问题是如何把基于文本的内容分列显示。通过上面的代码，为文本段落设置其中的列的数值，文本内容将会按你设定的值分成宽度相同的列。

本例添加以上样式后的效果如图 6.12 所示。



图 6.12 CSS 3 文本分列

其他 CSS 3 column 相关属性见表 6.1。

表 6.1 CSS 3 column 相关的属性

属性名	属性作用
column-count	可以定义栏目的数目
column-width	可以定义每栏的宽度
column-gap	定义两栏之间的间距距离
column-rule	定义每栏之间边框
column-rule-width	定义每栏之间边框的宽度
column-rule-style	定义每栏之间边框的样式（实线、虚线）
column-rule-color	定义每栏之间边框的颜色
column-span	定义元素可以在栏目上定位显示

6.15 让 div 层在 Flash 之上

在网页上放置了 Flash 之后，浮动层浮动到 Flash 上方时，会被 Flash 遮住，设置 z-index 属性也无济于事。事实上，这个问题可以通过设置 Flash 的控制层参数来解决。

在 Flash 里加入如下属性即可：

```
wmode = "opaque"
```

即在<object>及</object>之间加入代码：

```
<param name="wmode" value="opaque">
```

或在<embed>里加入如下代码：

```
wmode="opaque"
```

通过设置 wmode 为 opaque，可以使得目标 Flash 在页面上显示在最下面，这种做法可以使得其他元素都覆盖在这个 Flash 之上。完整的 CSS 代码如下：

```
01 * {
02     margin: 0px;
03     padding: 0px;
04 }
05 .flash {
06     height: 500px;
07     position: relative;
08 }
09 .main {
10     background: #000;
11     width: 500px;
12     height: 30px;
13     position: absolute;
14     left: 10px;
15     top: 30px;
16     z-index: 2;
17     color: #fff;
18 }
```

HTML 部分代码如下：

```
01 <div class="flash">
02     <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.
    macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0" width="1002" height="426">
03         <param name="movie" value="publish/example.swf" />
04         <param name="quality" value="high" />
05         <param name="wmode" value="transparent" />
06         <param name="wmode" value="opaque" />
07         <embed src="publish/example.swf" quality="high" wmode="transparent" pluginspage=
    "http://www.macromedia.com/go/getflashplayer" type="application/x-shockwave-flash" width="500"
    height="220"></embed>
08     </object>
```

```
09 <div class="main">
10 这是浮动在 Flash 上方的浮动层。
11 </div>
12 </div>
```

本例通过在 Flash 中加入语句`<param name="wmode" value="transparent" />`，对 Flash 的 wmode 属性设置为 transparent，从而达到了让 div 呈现在 Flash 之上的效果，显示效果如图 6.13 所示。



图 6.13 div 在 Flash 上方

关于 wmode 的属性的几个取值说明可见表 6.2。

表 6.2 比较wmode不同取值

属性取值	作用
wmode=window	在Web上用影片自己的矩形窗口来播放应用程序。 "Window"表明此Flash应用程序与HTML层没有任何交互，并且始终位于顶层。
wmode=opaque	使应用程序隐藏页面上位于它后面的所有内容。
wmode=transparent	使HTML页的背景可以透过应用程序的所有透明部分显示出来，可能会降低动画性能。
column-rule	定义每栏之间边框
column-rule-width	定义每栏之间边框的宽度
column-rule-style	定义每栏之间边框的样式（实线、虚线）

opaque 与 transparent 两种取值的区别是，当 wmode 取值为 transparent 时，允许 Flash 透明，即作为 Flash 的 src 值的 swf 文件是允许透明的，位于该 Flash 下层的元素可以透过透明部分显示给用户，而当 wmode 取值为 opaque 时，则不透明。

6.16 float 引起 div 自适应高度无效的解决方案

先看一实例：

```
//CSS 代码
01 .parent
02 {
03     border:2px solid red;
04     width:200px;
05 }
06 .children
07 {
08     width:200px;
09     height:200px;
10     background-color:green;
11 }
//HTML 代码
12 <div class="parent">
13     <div class="children"></div>
14 </div>
```

以上代码中，父元素只规定了宽度没有规定高度，子元素在正常的文档流中，所以子元素能够将父元素撑开，即父元素的高度可根据子元素的高度自动适应。

再看一实例：

```
//CSS 代码
01 .parent
02 {
03     border:2px solid red;
04     width:200px;
05 }
06 .children
07 {
08     width:200px;
09     height:200px;
10     float:left;
11     background-color:green;
12 }
//HTML 代码
13 <div class="parent">
14     <div class="children"></div>
15 </div>
```

从上面代码的运行可以看出，父元素并没有被撑开，即父元素并未根据子元素自动调节高度，这是因为子元素 `float` 属性值设置为 `left`，从而脱离了文档流。如果希望父元素达到高度自适应，就要清除子元素的浮动。修改代码如下：

```
//CSS 代码
01 .parent
02 {
03     border:2px solid red;
04     width:200px;
```

```

05     overflow:hidden;
06 }
07 .children
08 {
09     width:200px;
10     height:200px;
11     float:left;
12     background-color:green;
13 }
//HTML 代码
14 <div class="parent">
15     <div class="children"></div>
16 </div>

```

以上代码通过给父元素的样式中添加 `overflow:hidden` 即可清除浮动。如果说父元素像最开始的例子一样，固定了高度，并且还要实现高度自适应效果，就可以参阅 6.8 节设置最小高度的情况，解决方法是一致的。代码如下：

```

//CSS 代码
01 .parent
02 {
03     border:2px solid red;
04     width:200px;
05     height:auto!important;
06     height:100px;
07     min-height:100px;
08 }
09 .children
10 {
11     width:200px;
12     height:200px;
13     float:left;
14     background-color:green;
15 }
16 .clear
17 {
18     clear:both;
19 }
//HTML 代码
20 <div class="parent">
21     <div class="children"></div>
22     <div class="clear"></div>
23 </div>

```

在 6.12 节中介绍了升级版的 Clearfix，可以避免单独定义非语义化的节点 `<div class="clear"></div>`。因此，本例可以改写为：

```

//CSS 代码

```

```
01 .parent
02 {
03     border:2px solid red;
04     width:200px;
05     height:auto!important;
06     height:100px;
07     min-height:100px;
08 }
09 .children
10 {
11     width:200px;
12     height:200px;
13     float:left;
14     background-color:green;
15 }
16 .clearfix:after {
17     content: ".";
18     display: block;
19     height: 0;
20     clear: both;
21     visibility: hidden
22 }
23 .clearfix {
24     display: inline-block
25 }
26 * html .clearfix {
27     height: 1%
28 }
29 .clearfix {
30     display: block
31 }
//HTML 代码
32 <div class="parent clearfix">
33     <div class="children"></div>
34 </div>
```

其中，为父元素添加 class 属性 clearfix，使用升级版的 Clearfix 清除子元素的浮动。

6.17 Flexbox 布局风格

Flexbox 是一个用于页面布局的全新 CSS 3 模块功能，可以把列表放在同一个方向（从左到右或从上到下排列），并且让这些列表能延伸到所有可用的空间。Flexbox 布局的目标是提供更灵活的布局元素之间的空间，即使元素的大小是未知的或是动态变化的。Flexbox

布局的主体思想是使得元素可以改变大小以适应可用空间，当可用空间变大，Flex 元素将伸展大小以填充可用空间；当 Flex 元素超出可用空间时将自动缩小。

Flexbox 是布局模块而不是一个简单的属性，它包括父元素和子元素的属性。较为复杂的布局可以通过嵌套一个伸缩容器（flex container）来辅助实现。例如：

//CSS 代码

```
01 .container {
02     display: -webkit-flex;
03     display: flex;
04 }
05 nav {
06     width: 200px;
07 }
08 .flex-column {
09     -webkit-flex: 1;
10     flex: 1;
11 }
```

HTML 代码如下：

```
01 <div class="container elem">
02     <span class="label">&lt;div class="container"&gt;</span>
03
04     <nav class="elem elem-red">
05         <span class="label">&lt;nav&gt;</span>
06         <ul>
07             <li>
08                 <a href="flexbox.html">主页</a>
09             </li>
10             <li>
11                 <a href="flexbox.html">内容页</a>
12             </li>
13             <li>
14                 <a href="flexbox.html">详情页</a>
15             </li>
16         </ul>
17     <span class="endlabel">&lt;/nav&gt;</span>
18 </nav>
19
20 <div class="elem elem-red flex-column">
21     <span class="label">&lt;div class="flex-column"&gt;</span>
22
23     <section class="elem elem-green">
24         <span class="label">&lt;section&gt;</span>
25         <p>
26             Flexbox 好容易使用！
27         </p>
28     </section>
29 </div>
```

```

28     </p>
29     <span class="endlabel">&lt;/section&gt;</span>
30 </section>
31
32 <section class="elem elem-green ipsum">
33     <span class="label">&lt;section&gt;</span>
34     <p>
35         Flexbox 的布局是一个用于页面布局的全新 CSS 3 模块功能，可以把列表放在同一个
           方向……（省略部分文字），Flex 元素将伸展大小已填充可用空间。
36     </p>
37     <span class="endlabel">&lt;/section&gt;</span>
38 </section>
39
40 <span class="endlabel">&lt;/div&gt;</span>
41 </div>
42
43 </div>

```

显示效果如图 6.14 所示。

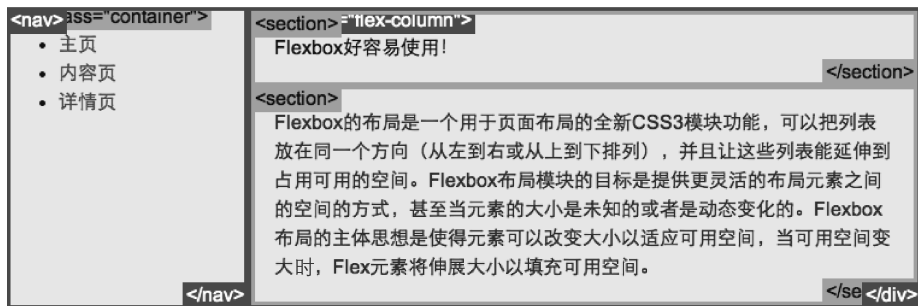


图 6.14 Flexbox 布局

Flexbox 还可以简单快速地创建一个具有弹性功能的布局，当在一个小屏幕上显示时，Flexbox 可以让元素在容器（伸缩容器）中进行自由扩展和收缩，从而容易调整整个布局。它的目的是使得常见的布局模式（如多列布局）可以非常简单地实现。例如：

```

//CSS 代码
01 .container {
02     display: -webkit-flex;
03     display: flex;
04 }
05 .initial {
06     -webkit-flex: initial;
07     flex: initial;
08     width: 200px;
09     min-width: 100px;
10 }
11 .none {

```



```

12  -webkit-flex: none;
13      flex: none;
14  width: 200px;
15  }
16  .flex1 {
17      -webkit-flex: 1;
18      flex: 1;
19  }
20  .flex2 {
21      -webkit-flex: 2;
22      flex: 2;
23  }
//HTML 代码
01  <div class="container elem">
02      <section class="elem elem-green initial">
03          <span class="label">&lt;div class="initial"&gt;</span>
04          <p>
05              空间足够的时候，我的宽度是 200px，如果空间不足，我会变窄到 100px，但不会再窄了。
06          </p>
07          <span class="endlabel">&lt;/div&gt;</span>
08      </section>
09
10      <section class="elem elem-green none">
11          <span class="label">&lt;div class="none"&gt;</span>
12          <p>
13              无论窗口如何变化，我的宽度一直是 200px。
14          </p>
15          <span class="endlabel">&lt;/div&gt;</span>
16      </section>
17
18      <section class="elem elem-green flex1">
19          <span class="label">&lt;div class="flex1"&gt;</span>
20          <p>
21              我会占满剩余宽度的 1/3。
22          </p>
23          <span class="endlabel">&lt;/div&gt;</span>
24      </section>
25
26      <section class="elem elem-green flex2">
27          <span class="label">&lt;div class="flex2"&gt;</span>
28          <p>
29              我会占满剩余宽度的 2/3。
30          </p>
31          <span class="endlabel">&lt;/div&gt;</span>
32      </section>

```

33 </div>

显示效果如图 6.15 所示。

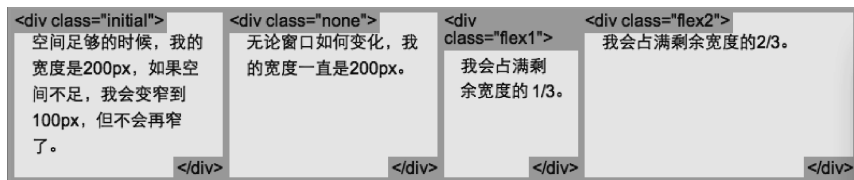


图 6.15 Flexbox 分栏布局

使用 Flexbox 居中布局实现如下：

//CSS 代码

```
01 .vertical-container {
02   display: -webkit-flex;
03   display: flex;
04   height: 300px;
05 }
06 .vertically-centered {
07   margin: auto;
08 }
```

//HTML 代码

```
01 <div class="vertical-container elem">
02   <span class="label">&lt;div class="vertical-container"&gt;</span>
03   <section class="elem elem-green vertically-centered">
04     <span class="label">&lt;div class="vertically-centered"&gt;</span>
05     <p>
06       CSS 里总算是有了一种简单的垂直居中布局的方法了！
07     </p>
08     <span class="endlabel">&lt;/div&gt;</span>
09   </section>
10   <span class="endlabel">&lt;/div&gt;</span>
11 </div>
```

显示效果如图 6.16 所示。



图 6.16 Flexbox 居中布局

6.18 动态高度下的居中

第 6.1~6.4 节中介绍了文本居中的方式，本节介绍动态高度下的居中方法，旨在设计当容器高度发生变化时，元素仍然保持垂直方向上的居中。具体实现代码如下：

```
//CSS 代码
01  *{
02      margin: 0;
03      padding: 0;
04  }
05  #page {
06      display: table;
07      overflow: hidden;
08      margin: 0px auto;
09  }
10  /*IE 7*/
11  *:first-child+html #page {
12      position: relative;
13  }
14  /*IE 6*/
15  * html #page {
16      position: relative;
17  }
18  #content_container {
19      display: table-cell;
20      vertical-align: middle;
21  }
22  /*IE 7*/
23  *:first-child+html #content_container {
24      position: absolute;
25      top: 50%;
26  }
27  /*IE 6*/
28  * html #content_container {
29      position: absolute;
30      top: 50%;
31  }
32  /*IE 7*/
33  *:first-child+html #content {
34      position: relative;
35      top: -50%;
36  }
37  /*IE 6*/
38  * html #content {
```

```

39     position: relative;
40     top: -50%;
41 }
42 html,body {
43     height: 100%;
44 }
45 #page {
46     height: 100%;
47     width: 465px;
48 }
//HTML 代码
49 <div id="page">
50     <div id="content_container">
51         <div id="content">
52             <p>your content</p>
53         </div>
54     </div>
55 </div>

```

其中，容器#page 的高度为 100%，#content 设置相对定位，且 top 为 50%。在代码第 14~17 行、第 27~31 行、第 37~41 行代码对 IE 6 做 Hack，在代码第 10~13 行、第 22~26 行、第 32~36 行代码对 IE 7 做 Hack，因此，本实例在 IE 6、IE 7 等浏览器上均兼容。

6.19 纯 CSS 实现固定表头

本节介绍固定表头的 CSS 表格，CSS 代码如下：

```

01  /*定义可滚动区域宽、高样式，其中，增加滚动条的 16px*/
02  div.tableContainer {
03      clear: both;
04      border: 1px solid #963;
05      height: 285px;
06      overflow: auto;
07      width: 756px
08  }
09  /* 定义 overflow 为 hidden */
10  html>body div.tableContainer {
11      overflow: hidden;
12      width: 756px
13  }
14  /* 定义表格宽度 */
15  div.tableContainer table {
16      float: left;
17      width: 740px

```

```
18 }
19 /*设置 thead 样式*/
20 html>body thead.fixedHeader tr {
21     display: block
22 }
23 /*定义 th 样式*/
24 thead.fixedHeader th {
25     background: #C96;
26     border-left: 1px solid #EB8;
27     border-right: 1px solid #B74;
28     border-top: 1px solid #EB8;
29     font-weight: normal;
30     padding: 4px 3px;
31     text-align: left
32 }
33 /*定义 a 元素样式*/
34 thead.fixedHeader a, thead.fixedHeader a:link, thead.fixedHeader a:visited {
35     color: #FFF;
36     display: block;
37     text-decoration: none;
38     width: 100%
39 }
40 /*a:hover 样式*/
41 thead.fixedHeader a:hover {
42     color: #FFF;
43     display: block;
44     text-decoration: underline;
45     width: 100%
46 }
47 /*定义表格内容可滚动样式，tbody 元素为块级元素*/
48 html>body tbody.scrollContent {
49     display: block;
50     height: 262px;
51     overflow: auto;
52     width: 100%
53 }
54 /*td 元素样式*/
55 tbody.scrollContent td, tbody.scrollContent tr.normalRow td {
56     background: #FFF;
57     border-bottom: none;
58     border-left: none;
59     border-right: 1px solid #CCC;
60     border-top: 1px solid #DDD;
61     padding: 2px 3px 3px 4px
62 }
```

```

63  /*th 样式*/
64  html>body thead.fixedHeader th {
65      width: 200px
66  }
67  html>body thead.fixedHeader th + th {
68      width: 240px
69  }
70  html>body thead.fixedHeader th + th + th {
71      width: 316px
72  }
73  /*td 元素样式*/
74  html>body tbody.scrollContent td {
75      width: 200px
76  }
77  html>body tbody.scrollContent td + td {
78      width: 240px
79  }
80  html>body tbody.scrollContent td + td + td {
81      width: 300px
82  }

```

此段代码省略了 reset 基本样式的代码段，reset 部分详见源码。首先定义固定表头的 CSS 表格的区域，即设置宽与高，其中宽度包含滚动条的宽度，并设置 overflow 为 hidden。代码第 21 行设置 thead 的 display 为 block，即定义其为块内元素。重点在于代码第 51 行，设置 scrollContent（即 table）内容的 overflow 属性为 auto，即可滚动。其他部分分别定义 th、a、td 元素的样式。

HTML 代码如下：

```

01  <div id="tableContainer" class="tableContainer">
02  <table border="0" cellpadding="0" cellspacing="0" width="100%" class="scrollTable">
03  <thead class="fixedHeader">
04  <tr class="alternateRow">
05  <th><a href="#">固定表头 1</a></th>
06  <th><a href="#">固定表头 2</a></th>
07  <th><a href="#">固定表头 3</a></th>
08  </tr>
09  </thead>
10  <tbody class="scrollContent">
11  <tr class="normalRow">
12  <td>固定表头的 CSS 表格 1</td>
13  <td>固定表头的 CSS 表格 2</td>
14  <td>固定表头的 CSS 表格 3</td>
15  </tr>
16  <tr class="alternateRow">
17  <td>固定表头的 CSS 表格 1</td>
18  <td>固定表头的 CSS 表格 2</td>

```

```
19      <td>固定表头的 CSS 表格 3</td>
20    </tr>
21  </tbody>
22 </table>
23 </div>
```

最终展示效果如图 6.17 所示。

固定表头 1	固定表头 2	固定表头 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3
固定表头的CSS表格 1	固定表头的CSS表格 2	固定表头的CSS表格 3

图 6.17 响应式多数据列表框

6.20 Metro 布局风格

Metro 是微软在 Windows Phone 7 中正式引入的一种界面设计语言，也是 Windows 8 的主要界面显示风格。Metro 布局风格的特点是基于平面的排版，具有干净和整洁的视觉效果。本节介绍使用 CSS 来模拟 Metro 布局风格。

HTML 代码如下：

```
//HTML 代码
01 <div id="place">
02   <div id="name1">
03     Metro 风格布局
04   </div>
05   <div id="wrapper1">
06     <div id="thumb1-1">
07     </div>
08     <div id="thumb1-2">
09     </div>
10     <div id="thumb1-3">
11     </div>
12     <div id="thumb1-4">
13     </div>
14     <div id="thumb1-5">
15     </div>
```

```

16     <div id="thumb1-6">
17     </div>
18     <!--..... 省略部分代码，具体请参考源代码-->
19     <form action="" method="get">
20         <input type="text" name="q" value=""/><button type="submit"></button>
21         <div id="engines1">
22             <div id="google1">
23             </div>
24             <div id="bing1">
25             </div>
26             <div id="yahoo1">
27             </div>
28             <div id="wikipedia1">
29             </div>
30         </div>
31         <div id="search-engine1">
32         </div>
33     </form>
34 </div>
35 </div>

```

CSS 代码如下：

```

01  /*
02  * reset 全局样式，去除元素默认内外边距、边框
03  * 设置全局字体为 sans-serif，字体大小为 1em
04  */
05  * {
06      margin: 0;
07      padding: 0;
08      border: 0;
09      font-family: sans-serif;
10      font-size: 1em;
11      font-weight: normal;
12      font-style: normal;
13      text-decoration: none;
14  }
15  /* 设置背景颜色 */
16  body {
17      background: #260930;
18      overflow: hidden;
19  }
20  /* 设置整个页面的容器样式，相对定位 */
21  #place {
22      position: relative;
23      width: 3065px;

```



```

24     height: 560px;
25     margin: 0px;
26 }
27 /* 设置一页 Metro 的容器样式，绝对定位*/
28 #wrapper1{
29     width: 975px;
30     height: 480px;
31     position: absolute;
32     top: 80px;
33 }
34 /* 设置 form 样式，包括宽、高、字体等样式 */
35 form {
36     width: 635px;
37     height: 30px;
38     position: absolute;
39     left: 340px;
40     top: 225px;
41     font-family: Tahoma, Arial, "MS Trebuchet", sans-serif;
42     text-align: center;
43     padding: 0px;
44 }
45 /* 设置输入框样式，包括宽、高、背景颜色、边框、字体、字体颜色、字体大小等样式 */
46 input {
47     width: 583px;
48     height: 24px;
49     margin: 5px 0px 5px 5px;
50     background: #fff;
51     border: 2px #fff solid;
52     color: #333;
53     font-family: Tahoma, Arial, "MS Trebuchet", sans-serif;
54     font-size: 14px;
55     line-height: 24px;
56     padding: 1px;
57 }
58 /* 设置 Metro 一个颜色单元的定位、背景颜色、宽、高等样式*/
59 #thumb1-1{
60     width: 215px;
61     height: 103px;
62     position: absolute;
63     left: 0px;
64     top: 0px;
65     background: #409da5;
66     line-height: 103px;
67 }
68 /* .....省略部分代码，具体请参考源代码 */

```

Metro 布局设计的核心思想类似于拼图游戏，两个或者多个小矩形一边宽度之和与另一个大矩形的一边宽度相等，由此形成块状结构，且相邻矩形块的背景颜色必不相同，形成错落有致的风格。其 HTML 结构包括一个大容器包裹 Metro 布局，每个 Metro 单元又由一个容器包裹，且设置为相对定位。每个矩形块设置为绝对定位，再各自计算各个矩形所需设置的 left 值与 top 值。

本例中，HTML 结构中还包含有 form 表单，代码第 34~57 行设计表单的样式，表单也为矩形。代码第 58~67 行即为矩形块的样式示例，设置了矩形块的宽度、高度、left 值、top 值、背景颜色，并设置为绝对定位。其他矩形块的样式与此矩形块的设置方法类似，这里就不再赘述，详见源代码。

最终展示效果如图 6.18 所示。



图 6.18 Metro 风格布局

第 7 章 美化与装饰



网站上很大一部分内容是通过文本直接展示的，对文本进行合理的美化，能给用户带来良好的视觉体验。本章将重点讲述如何对文本、输入框、列表、翻页页码、链接等页面元素进行美化与装饰。

本章主要涉及的内容有：

- 文本的装饰、锚链接装饰、输入框发光与高亮美化
- 自定义滚动条、分隔线、列表符号
- 跨浏览器的透明度
- 鼠标指向特效
- 翻页页码不同风格的设计与实现
- 页面局部特色效果，如顶部阴影、页面卷曲、针线缝合效果

7.1 文本装饰

在网页制作的过程中，常常需要对文本进行装饰以提高用户体验。本节主要探讨文本的颜色、文本画线、文本空白、文本方向等 4 个方面。

7.1.1 文本的颜色

装饰文本的颜色可由 `color` 属性进行设置。`color` 又称为 CSS 的前景色，通常使用在文字上。`color` 属性取值可为具体的颜色值，也可为 `inherit`。`color` 属性具有继承性，即对于父元素定义的 `color` 会延伸到子元素中。

```
//CSS 代码
01 body {
02     color: red
03 }
04 h1 {
05     color: #00ffcc
06 }
07 p.ex {
08     color: rgb(0,0,255)
09 }
```

```
//HTML 代码
10 <body>
11     <h1>超实用的 CSS 代码段</h1>
12     <div>超实用的 CSS 代码段超实用的 CSS 代码段超实用的 CSS 代码段超实用的 CSS 代码
13 段</div>
14     <p class="ex">超实用的 CSS 代码段</p>
15 </body>
```

展示效果如图 7.1 所示。

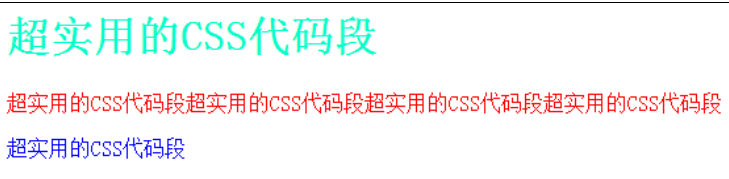


图 7.1 文本颜色

其中，在 CSS 代码中的第 2 行，color 取值为颜色名称。第 5 行 color 取值为 16 进制值。第 8 行 color 取值为 rgb 代码。第 2 行对父元素定义的颜色延伸到子元素中，也即子元素 div 继承了父元素 body 的颜色值。

7.1.2 文本画线

text-decoration 属性可以对文本设置特殊效果，如上画线、下画线等。目前，所有主流浏览器都支持 text-decoration 属性，但是 IE 浏览器不支持该属性的继承，即对父元素定义的 text-decoration 效果不会延伸到子元素中，子元素所需的效果需重新定义。表 7.1 列出了 text-decoration 可能的取值，并对不同取值与浏览器支持的情况进行了说明。

表 7.1 text-decoration的取值及浏览器支持情况

属性值	属性值的作用	浏览器支持情况
none	默认值，关闭文本装饰	所有主流浏览器支持
underline	定义文本下的一条线	所有主流浏览器支持
overline	定义文本上的一条线	所有主流浏览器支持
line-through	定义穿过文本的一条线	所有主流浏览器支持
blink	定义闪烁的文本	IE/Chrome/Safari不支持
inherit	定义从父元素继承该属性值	IE不支持

具体的文本装饰实例代码如下：

```
//CSS 代码
01 body {
02     background: black;
03     color: lime;
04 }
05 h1 {
06     text-decoration: overline;    /*定义文本上的一条线*/
```

```

07 }
08 h2 {
09     text-decoration: line-through;    /*定义贯穿文本的一条线*/
10 }
11 h3 {
12     text-decoration: underline;      /*定义文本下的一条线*/
13 }
14 h4 {
15     text-decoration: blink;          /*定义文本闪烁效果*/
16 }
17 a {
18     text-decoration: none;           /*去除文本装饰效果*/
19 }
//HTML 代码
20 <h1>超实用的 CSS 代码段</h1>
21 <h2>超实用的 CSS 代码段</h2>
22 <h3>超实用的 CSS 代码段</h3>
23 <h4>超实用的 CSS 代码段</h4>
24 <p>
25     <a href="http://www.css.com/index.html">超实用的 CSS 代码段</a>
26 </p>

```

第 6 行代码，使用 `overline` 属性在 h1 元素的文本上方设置了一条线。第 9 行代码，使用 `line-through` 属性对 h2 元素设置了贯穿文本的一条线。第 12 行代码，使用 `underline` 属性在 h3 元素的文本下方设置了一条线。第 15 行代码，使用 `blink` 属性对 h4 元素设置文本闪烁。由于 a 标签的默认样式是在文本下方有一条线，所以第 18 行代码使用 `none` 属性取消 a 标签下方的默认直线，但是并不会取消文本的 `color` 效果。本例展示效果如图 7.2 所示。



图 7.2 文本装饰

可以在规则中组合多种装饰效果。例如，需要对 h3 元素同时添加上画线和下画线：

```
//CSS 代码
```

```
01 h3 {
02     text-decoration: underline overline;
03 }
//HTML 代码
<h3>超实用的 CSS 代码段</h3>
```

h3 元素同时添加上画线和下画线的效果如图 7.3 所示。



图 7.3 添加上画线和下画线

7.1.3 文本的空白

文本的空白可通过 white-space 属性进行设置。通过设置这个属性，可以处理文本之间的空白符。默认情况下所有空白符会合并为一个空格。例如，有如下代码，在浏览器中显示各个字之间只会显示一个空格，同时忽略文本的换行。

```
//HTML 代码
01 <p>超 实 用 的 CSS
02 代 码 段</p>
```

white-space 的取值见表 7.2。

表 7.2 white-space的取值

属性值	说明	浏览器支持情况
normal	默认取值。空白会被浏览器忽略，多个空白符合并为一个空格	
pre	保留空白。其行为方式类似<pre>标签	IE7及IE7以下版本不支持
nowrap	文本不换行，在同一行上继续，直到遇到 为止	
pre-wrap	保留空白符序列，正常地进行换行	IE7及IE7以下版本不支持
pre-line	合并空白符序列，保留换行符	IE7及IE7以下版本不支持
inherit	规定应该从父元素继承white-space属性的值	

7.1.4 文本的方向

文本的方向可通过 direction 属性进行设置，其可取的属性值参见表 7.3。

表 7.3 direction的取值

属性值	说明
ltr	默认取值。文本方向从左到右
rtl	文本方向从右到左
inherit	规定应该从父元素继承 direction 属性的值

例如，普什图、阿拉伯、希伯来等国家的语言通常从右到左显示，需加入以下 CSS 代码：

```
//CSS 代码
01 html {
02     direction: rtl
03 }
```

其中，第2行设置文本方向为从右到左显示，效果如图7.4所示。

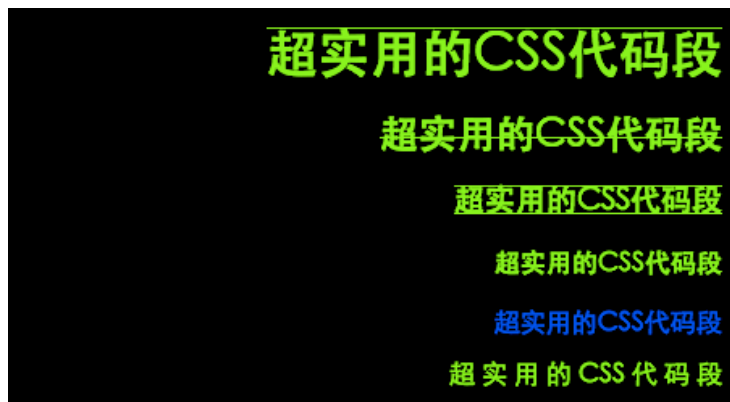


图 7.4 定义文本从右到左显示

7.2 发光输入框

在 Safari 浏览器下，任何输入框默认都会有一个发光的蓝色边框。如果想在其他浏览器上实现这个效果，可使用 CSS 3 技巧。需要注意的是，这并不是单独使用 box-shadow 属性来实现的，该发光输入框是向四周均匀发光，包含一个缓慢发光过渡效果。



图 7.5 发光输入框

发光输入框的实现代码如下：

```
//HTML 代码
01 <form>
02     <div>
03         <label for="username">账号:
04             <input name="username" type="text" placeholder="请输入账号">
05         </label>
06     </div>
07     <div>
08         <label for="password">密码:
```

```

09         <input name="password" type="password" placeholder="请输入密码"></label>
10     </div>
11     <div>
12         <label for="note">留言:
13         <textarea name="note" placeholder="请输入留言"></textarea>
14     </label>
15 </div>
16
17 </form>

```

CSS 代码如下:

```

01 input, textarea {
02     -webkit-transition: all 0.30s ease-in-out;
03     -moz-transition: all 0.30s ease-in-out;
04     -ms-transition: all 0.30s ease-in-out;
05     -o-transition: all 0.30s ease-in-out;
06     outline: none;
07     padding: 3px 0px 3px 3px;
08     margin: 5px 1px 3px 0px;
09     border: 1px solid #ddd;
10 }
11 input:focus, textarea:focus {
12     box-shadow: 0 0 5px rgba(81, 203, 238, 1);
13     padding: 3px 0px 3px 3px;
14     margin: 5px 1px 3px 0px;
15     border: 1px solid rgba(81, 203, 238, 1);
16 }

```

在 CSS 代码中, 分别定义了输入框在获得焦点前与获得焦点时的效果。其中, 第 15 行定义了输入框边框的宽度、线条类型、边框颜色, 第 12 行通过 `box-shadow` 属性定义输入框的边框阴影。第 2~5 行通过 `transition` 属性定义了发光的变化过程, 并且在 `transition` 属性前面加上 `-webkit-`、`-moz-`、`-ms-`、`-o-` 等属性前缀, 支持不同浏览器类型。

7.3 自定义滚动条

在 6.13 节中, 介绍了强制显示或隐藏滚动条的方法, 但是, 显示的滚动条仍然是默认的滚动条样式。目前支持自定义滚动条样式的有 IE 浏览器、Webkit 内核浏览器, 两者设定方法不同。

在 IE 浏览器下自定义滚动条的实例代码如下:

```

//CSS 代码
01 body{
02     scrollbar-arrow-color: red;           /*上下按钮上三角箭头的颜色*/
03     scrollbar-face-color: #cbcbcb;       /*滚动条凸出部分的颜色*/

```



```

04 scrollbar-3dlight-color: blue;      /*滚动条亮边的颜色*/
05 scrollbar-highlight-color: #333;    /*滚动条空白部分的颜色*/
06 scrollbar-shadow-color: yellow;     /*滚动条阴影的颜色*/
07 scrollbar-darkshadow-color: green;  /*滚动条强阴影的颜色*/
08 scrollbar-track-color: #eee;        /*滚动条背景颜色*/
09
10 scrollbar-base-color: black;        /*滚动条的基本颜色*/
11 Cursor:url(mouse.cur);              /*自定义个性鼠标*/
12 /*以上 2 项适用与: body、div、textarea、iframe*/
13 }

```

其中, scrollbar-base-color 与 Cursor 属性适用于元素 body、div、textarea、iframe。在 IE 下定义的滚动条样式无法被其他浏览器识别。具体的颜色值可根据设计方案进行调整, 其在 IE 下的展示效果如图 7.6 所示。

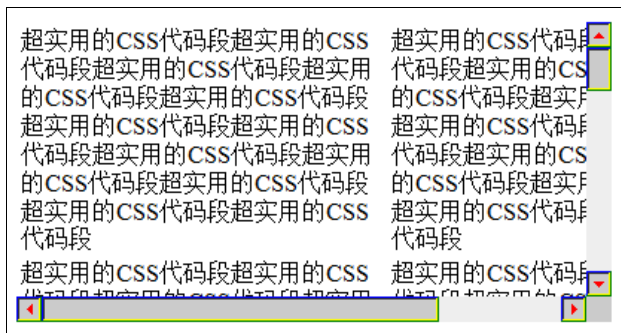


图 7.6 IE 自定义滚动条

在 Chrome 浏览器上自定义滚动条的实例代码如下:

```

//HTML 代码
03 <div class="demo">
04     <table width="500" border="0">
05         <tr>
06             <td>超实用的 CSS 代码段</td>
07             <td>超实用的 CSS 代码段</td>
08         </tr>
09         <tr>
10             <td>超实用的 CSS 代码段</td>
11             <td>超实用的 CSS 代码段</td>
12         </tr>
13         <tr>
14             <td>超实用的 CSS 代码段</td>
15             <td>超实用的 CSS 代码段</td>
16         </tr>
17         <tr>
18             <td>超实用的 CSS 代码段
19 段</td>

```

```

20         </tr>
21         <tr>
22             <td>超实用的 CSS 代码段</td>
23             <td>超实用的 CSS 代码段</td>
24         </tr>
25         <tr>
26             <td>超实用的 CSS 代码段</td>
27             <td>超实用的 CSS 代码段</td>
28         </tr>
29     </table>
30 </div>

```

CSS 代码如下：

```

01 ::-webkit-scrollbar {                /* 滚动条整体部分 */
02     width:10px;
03     margin-right:2px
04 }
05 ::-webkit-scrollbar-button {         /* 滚动条两端的按钮 */
06     width:10px;
07     background-color: yellow;
08 }
09 ::-webkit-scrollbar:horizontal {
10     height:10px;
11     margin-bottom:2px
12 }
13 ::-webkit-scrollbar-track {          /* 外层轨道 */
14     border-radius: 10px;
15 }
16 ::-webkit-scrollbar-track-piece {    /* 内层轨道，滚动条中间部分 */
17     background-color: #333;
18     border-radius: 10px;
19 }
20 ::-webkit-scrollbar-thumb {          /* 滑块 */
21     width:10px;
22     border-radius: 5px;
23     background: #CBCBCB;
24 }
25 ::-webkit-scrollbar-corner {         /* 边角 */
26     width: 10px;
27     background-color: red;
28 }
29 ::-webkit-scrollbar-thumb:hover {    /* 鼠标移入滑块 */
30     background: #909090;
31 }
32 .demo {

```

```
33     width: 400px;
34     height: 200px;
35     overflow: auto;
36 }
```

在 Webkit 内核浏览器下，使用一系列的伪元素来实现自定义滚动条。其中一些伪元素的设置不是必需的，可根据需要进行选择性设置。这里为了展示详细的属性设置，对::-webkit-scrollbar-button、::-webkit-scrollbar-corner 等伪类均做了定义。所示效果如图 7.7。除了在以上代码段中所列出的伪元素外，Webkit 还提供了一系列的伪类，可以丰富滚动条的样式，详情参见表 7.4。

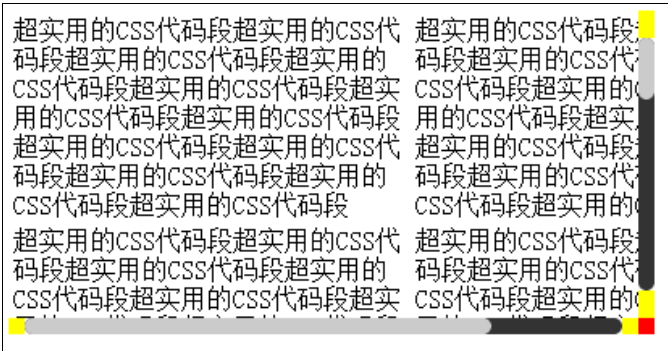


图 7.7 Chrome 自定义滚动条

表 7.4 Webkit 伪类

属性值	说明
:horizontal-horizontal	应用于水平方向的滚动条
:vertical-vertical	应用于垂直方向的滚动条
:decrement-decrement	应用于按钮和内层轨道，用来指示按钮或者内层轨道是否会减小视窗的位置，如垂直滚动条的上面、水平滚动条的左边
:increment-increment	用于指示按钮或内层轨道是否会增大视窗的位置，如垂直滚动条的下面、水平滚动条的右边
:start-start	应用于按钮和滑块，用于定义对象是否放到滑块的前面
:end-end	类似于start伪类，标识对象是否放到滑块的后面
:double-button	该伪类用于按钮和内层轨道。用来判断一个按钮是不是放在滚动条同一端的一对按钮中的一个。对于内层轨道来说，它表示内层轨道是否紧靠一对按钮
:single-button	类似于double-button伪类。对按钮来说，它用于判断一个按钮是否自己独立的在滚动条的一端。对内层轨道来说，它表示内层轨道是否紧靠一个single-button
:no-button	用于内层轨道，表示内层轨道是否要滚动到滚动条的终端，比如，滚动条两端没有按钮的时候
:corner-present	用于所有滚动条轨道，指示滚动条圆角是否显示
:window-inactive	用于所有的滚动条轨道，指示应用滚动条的某个页面元素当前是否被激活

说明：除了表 7.4 所列出的伪类之外，:enabled、:disabled、:hover 和:active 等伪类同样可以用于滚动条中。

7.4 页面顶部阴影

在网页制作的过程中，还可以对页面顶部进行阴影设置。实例代码如下：

```
//CSS 代码
01 body: before {
02     content: "";
03     position: fixed;
04     top: -10px;
05     left: 0;
06     width: 100%;
07     height: 10px;
08     box-shadow: 0px 0px 10px rgba(0,0,0,.8);
09     z-index: 100;
10 }
```

代码中使用伪元素:before 设置页面顶部阴影。:before 伪元素允许在元素内容的最前面插入生成内容。默认情况下:before 伪元素是行内元素，不过可以使用属性 display 改变。本例中，使用:before 在 body 之前加入内容，以生成阴影效果。第 8 行使用 box-shadow 属性定义阴影效果。目前，所有主流浏览器都支持:before 伪元素，IE9+、Firefox 4、Chrome、Opera 以及 Safari 5.1.1。该页面顶部阴影效果如图 7.8 所示。

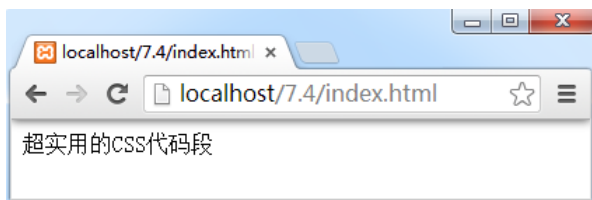


图 7.8 页面顶部阴影

7.5 巧妙实现分隔线

分隔线的实现方法有很多种，以下代码实例展示了几种常见的分隔线实现方法。

```
//CSS 代码
01 .demo{
02     width: 600px;
03 }
04 .line_01{
05     padding: 0 20px 0;
06     margin: 20px 0;
07     line-height: 1px;
08     border-left: 200px solid #ddd;
09     border-right: 200px solid #ddd;
```

```
10     text-align: center;
11 }
12 .line_02{
13     height: 1px;
14     border-top: 1px solid #ddd;
15     text-align: center;
16 }
17 .line_02 span{
18     position: relative;
19     top: -12px;
20     background: #fff;
21     padding: 0 20px;
22 }
23 .line_03{
24     width:600px;
25 }
26 .line_03 b{
27     background: #ddd;
28     margin-top: 4px;
29     display: inline-block;
30     width: 180px;
31     height: 1px;
32     _overflow: hidden;
33     vertical-align: middle;
34 }
35 .line_03 span{
36     display: inline-block;
37     width: 220px;
38     vertical-align: middle;
39     text-align: center;
40 }
41 .line_04{
42     width:600px;
43 }
44 .line_04{
45     overflow: hidden;
46     _zoom: 1;
47 }
48 .line_04 b{
49     background: #ddd;
50     margin-top: 12px;
51     float: left;
52     width: 26%;
```

```

53     height: 1px;
54     _overflow: hidden;
55 }
56 .line_04 span{
57     padding: 0 10px;
58     width: 32%;
59     float: left;
60     text-align: center;
61 }
62 .line_05{
63     letter-spacing: -1px;
64     color: #ddd;
65 }
66 .line_05 span{
67     letter-spacing: 0;
68     color: #222;
69     margin: 0 20px;
70 }
//HTML 代码
71 <div class="line_01">超实用的 CSS 代码段 单标签实现</div>
72 <br>
73 <br>
74 <div class="line_02"><span>超实用的 CSS 代码段 巧用色实现</span></div>
75 <br>
76 <br>
77 <div class="line_03"><b></b><span>超实用的 CSS 代码段 inline-block 实现
78 </span><b></b></div>
79 <br>
80 <br>
81 <div class="line_04"><b></b><span>超实用的 CSS 代码段 浮动实现</span><b></b></div>
82 <br>
83 <br>
84 <div class="line_05">—————<span>超实用的 CSS 代码段 字符实现</span>
85 —————</div>
86 <br>
87 <br>
88 </div>

```

第 4~11 行代码展示了仅使用一个标签即可实现分隔线的方法。第 17~25 行展示了使用背景色实现分隔线的方法。第 26~40 行代码展示了使用 inline-block 属性实现的分隔线方法。第 41~61 行代码展示了使用浮动实现分隔线的方法，缺点是需要定制元素的宽度。第 62~70 行代码展示了使用字符实现分隔线的方法，缺点是需要手动输入字符，会增大页面的体积。分隔线的效果如图 7.9 所示。

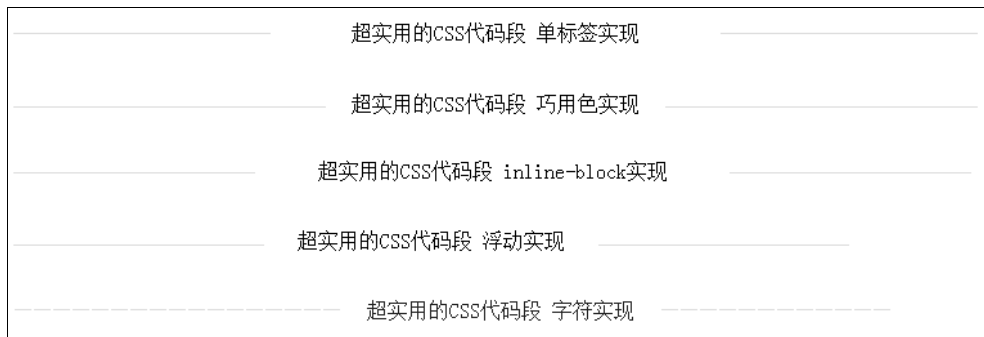


图 7.9 分隔线

7.6 三角形列表符号

通过 CSS 可以实现三角形的图案，该图案可用于列表符号。实现代码如下：

```
//CSS 代码
01  ul{
02      list-style: none;
03      width: 300px;
04  }
05  ul li {
06      border: 1px solid #eee;
07      height: 32px;
08      vertical-align: middle;
09      border-bottom: none;
10
11  }
12  ul li:last-child{
13      border-bottom: 1px solid #eee;
14  }
15  ul li .content{
16      display: inline-block;
17      height: 32px;
18      line-height: 32px;
19      vertical-align: middle;
20
21  }
22  .trigon {
23      display: inline-block;
24      float: left;
25      border-top: 5px solid #FFFFFF;
26      border-left: 5px solid #FF3300;
27      border-bottom: 5px solid #FFFFFF;
28  }
```

```
//HTML 代码
29 <ul>
30   <li>
31     <div class="trigon"></div>
32     <span class="content">列表一</span>
33   </li>
34   <li>
35     <div class="trigon"></div>
36     <span class="content">列表二</span>
37   </li>
38   <li>
39     <div class="trigon"></div>
40     <span class="content">列表三</span>
41   </li>
42   <li>
43     <div class="trigon"></div>
44     <span class="content">列表四</span>
45   </li>
46 </ul>
```

第 25~27 行中，通过设置 border-top、border-left、border-bottom 的值，使得元素呈现左三角形的形式。三角形列表符号的效果如图 7.10 所示。



图 7.10 三角形列表符号

7.7 纸页面卷曲效果

在页面上使用纸张卷曲效果，能呈现出更为接近真实书本的显示效果，提升用户阅读体验。实现代码如下：

```
//CSS 代码
01 body {
02   padding-top: 2.5em;
03   background-color: #666;
04   color: #333;
05   font-size: 84%;
06   font-family: "微软雅黑", "Yahei Mono";
07 }
01 a {
```



```
02     color: #fff;
03     position: absolute;
04     right: 1%;
05     top: 5px;
06 }
07 .title {
08     margin: 15px 0;
09     text-align: center;
10 }
11 .page {
12     width: 600px;
13     line-height: 30px;
14     margin: 0 auto;
15     padding: 15px 0 29px;
16     background-color: #f4f39e;
17     -moz-border-radius-bottomleft: 20px 500px;
18     -moz-border-radius-bottomright: 500px 30px;
19     -moz-border-radius-topright: 5px 100px;
20     -webkit-border-bottom-left-radius: 20px 500px;
21     -webkit-border-bottom-right-radius: 500px 30px;
22     -webkit-border-top-right-radius: 5px 100px;
23     border-bottom-left-radius: 20px 500px;
24     border-bottom-right-radius: 500px 30px;
25     border-top-right-radius: 5px 100px;
26     background:
27         -moz-repeating-linear-gradient(
28             top,
29             #fcf59b,
30             #fcf59b 29px,
31             #81cbbc 30px
32         );
33     background:
34         -webkit-gradient(
35             linear,
36             left top, left bottom,
37             from(#81cbbc),
38             color-stop(2%, #fcf59b)
39         );
40     background:
41         repeating-linear-gradient(
42             top,
43             #fcf59b,
44             #fcf59b 29px,
45             #81cbbc 30px
46         );
```

```

47     -webkit-background-size: 100% 30px;
48     -moz-box-shadow: 0 2px 10px 1px rgba(0, 0, 0, 0.2);
49     -webkit-box-shadow: 0 2px 10px 1px rgba(0, 0, 0, 0.2);
50     box-shadow: 0 2px 10px 1px rgba(0, 0, 0, 0.2);
51     text-shadow: 0 1px 0 #F6EF97;
52     position: relative;
53 }
54 .page p {
55     margin: 30px 0;
56     padding: 0 2em;
57 }
//HTML 代码
58 <div class="page">
59     <h2 class="title">一些示例文字</h2>
60     <p>纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果</p>
61     <p>纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果纸
62 张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果</p>
63     <p>今天纸张卷曲效果。。。</p>
64     <p>“纸张卷曲效果纸张卷曲效果纸张卷曲效果纸张卷曲效果” </p>
65 </div>

```

实现纸页面卷曲效果最重要的代码是第 23~25 行设定的 `border-bottom-left-radius`、`border-bottom-right-radius`、`border-top-right-radius`。第 20~22 行，为了支持 Chrome 浏览器加上 `-webkit-` 前缀，第 17~19 行为支持 Firefox 浏览器加上前缀 `-moz-`，且合并后两项 `bottom` 与 `left`，如 `-moz-border-radius-bottomleft`、`-moz-border-radius-bottomright`，该属性所带的两个参数，分别为水平圆角半径及垂直圆角半径，当水平圆角半径与垂直圆角半径相等时，呈现普通圆角效果，当水平圆角半径与垂直圆角半径不等时，呈现的是不规则圆角，纸页面卷曲效果正是利用不规则圆角并设置合适的半径大小而实现的。实现效果如图 7.11 所示。

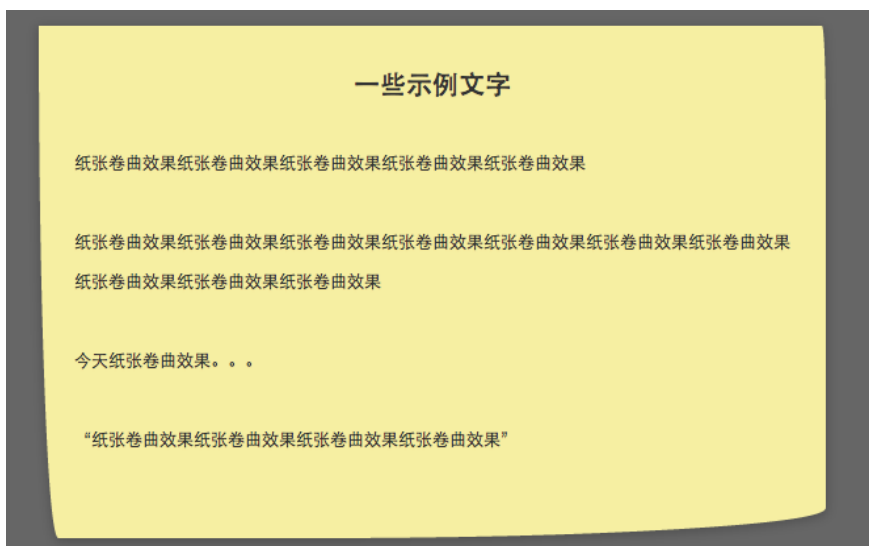


图 7.11 纸页面卷曲效果

7.8 跨浏览器的透明度

传统透明度的实现是通过图片进行模拟。随着浏览器的更新升级，通过为各个浏览器设定不同的属性，就可以设置跨浏览器的透明度了。实现代码如下：

```
//CSS 代码
01 body {
02     padding: 2em;
03     color: #ccc;
04     background: #000;
05 }
11 #out{
12     position:relative;
13     padding:20px 0;
14     height:100px;
15     width:200px;
16     color:#000;
17     border:2px solid #ddd;
18     _overflow:hidden;
19 }
20 #in{
21     background:#fff;
22     margin:0 10px;
23 }
24 #alpha{
25     position: absolute;
26     top:0;
27     left: 0;
28     width: 100%;
29     height:100%;
30     -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=30)";
31                                     /*IE 8*/
32     filter:alpha(opacity=30);      /*IE 5、IE 5.5、IE 6、IE 7*/
33     opacity: .3;                   /*Opera 9.0+、Firefox 1.5+、Safari、Chrome*/
34     *z-index: -1;                  /*让其位于 in 的下面*/
35     background:#fff;
36 }
//HTML 代码
37 <div id="out">
38     <div id="in">不透明<div>
39     <div id="alpha">半透明<div>
40 </div>
```

实现跨浏览器透明效果如图 7.12 所示。



图 7.12 跨浏览器半透明效果

这种写法的缺点是对父元素设置透明会导致子元素也随之被透明。以上代码使用兄弟节点避免这种情况的发生。这种改变 DOM 结构的解决方法，虽然实现起来不难，但是，改变了 DOM 结构不符合语义化的要求。因此，通过 CSS 3 的支持，进行改进后的代码如下：

```
//CSS 代码
01 .transparent{
02     background:rgba(0,0,0,0.5);          /* Chrome, Saf, FF, Opera <em>/
03     filter:
        progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,endColorstr=#990000
        00);
04                                           /* IE6 - IE9 */
05 }
```

其中 filter 可设置背景透明，参数 startColorStr 是可选项，设置或检索色彩渐变的开始颜色和透明度，其格式为 #AARRGGBB，AA、RR、GG、BB 为十六进制正整数。取值范围为 00-FF，AA 指定透明度，00 是完全透明，FF 是完全不透明，RR 指定红色值，GG 指定绿色值，BB 指定蓝色值，超出取值范围的值将被恢复为默认值，默认值为 #FF0000FF，不透明蓝色，取值范围为 #FF000000-#FFFFFFF。EndColorStr 是可选项，设置或检索色彩渐变的结束颜色和透明度，默认值为 #FF000000，不透明黑色。Enabled 也是可选项，取值为布尔值即 true 或 false，设置或检索滤镜是否激活。GradientType 是可选项，取值为整数值 0 或 1，设置或检索色彩渐变的方向，当取值为 0 时，色彩渐变方向为纵向，当取值为 1 时，色彩渐变方向为横向，默认取值为 0，即纵向。改进后的跨浏览器透明度设置代码如下：

```
//CSS 代码
01 .transparent{
02     background-image: -moz-linear-gradient(top,#ffffff,#000000);          /* FF3.6+ */
03     background-image: -webkit-gradient(linear, left top, left bottom, from(#ffffff), to(#000000));
04                                           /* Saf4+, Chrome */
05     background-image: -webkit-linear-gradient(top, #ffffff, #000000); /* Chrome 10+, Saf5.1+, iOS 5+ */
06     background-image: -ms-linear-gradient(top,#ffffff,#000000);          /* IE10 */
07     background-image: -o-linear-gradient(top,#ffffff,#000000);           /* Opera 11.10+ */
```

```
08     background-image: linear-gradient(top, #ffffff, #000000);
09     filter:    progid:DXImageTransform.Microsoft.gradient(startColorstr=#ffffff,endColorstr=
                #ff000000,enabled=true,gradientType='1');                /* IE6-IE9 */
10 }
```

其中，IE 滤镜可以实现渐变加透明，CSS 3 中将#000000 写法改为 rgba(0,0,0,0.5)即可。

7.9 鼠标指向时变成手型

默认情况下，鼠标经过链接 a 标签时，鼠标形状会变成手型。如果希望网页中其他元素在鼠标经过时，也会变成手型，可通过 CSS 的 cursor 属性进行设置。cursor 的取值及含义参见表 7.5。在 IE 下，父元素的 cursor 属性样式不会被子元素继承。

表 7.5 cursor的取值

属性值	说明
url	根据用户定义的资源显示
auto	默认值，显示为浏览器设置的光标
default	默认光标，通常是一个箭头
crosshair	十字形光标
pointer	手形光标
move	指示某对象可被移动
text	指示文字
wait	等待，指示程序正忙
help	指示可用的帮助
inherit	继承父元素的cursor属性值
progress	指示过程
e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize	指示某对象可被改变大小，可改变的方向不同。 e-resize右箭头形，n-resize 上箭头形，nw-resize左上 箭头形，w-resize左箭头形，s-resize下箭头形，se-resize 右下箭头形，sw-resize 左下箭头形

cursor 的默认取值为 auto，即根据浏览器的默认样式显示鼠标形状。为了让特殊的元素在鼠标经过时鼠标变成手型，可使用以下代码实现：

```
//CSS 代码
01 a[href], input[type='submit'], input[type='image'], label[for], select, button, .pointer {
02     cursor: pointer;
03 }
```

7.10 鼠标移动到 div 上高亮显示

7.9 节介绍了当鼠标移动到特定元素上时，鼠标形状可以发生相应的变化，以提醒用户该元素的特殊作用。但是，仅仅鼠标形状的变化有时仍然不能满足设计上对醒目的特效要求。本节将介绍当鼠标移动到 div 上时，高亮显示整个 div 的特效。实现代码如下：

//HTML 代码

```

01 <div class="pricing_table">
02   <ul>
03     <li>超实用的 CSS 代码段</li>
04     <li></li>
05     <li>超实用的 CSS 代码段</li>
06     <li>超实用的 CSS 代码段</li>
07     <li>超实用的 CSS 代码段</li>
08     <li>超实用的 CSS 代码段</li>
09     <li></li>
10     <li><a href="" class="buy_now">Buy Now</a></li>
11   </ul>
12   <ul>
13     <li>超实用的 CSS 代码段</li>
14     <li></li>
15     <li>超实用的 CSS 代码段</li>
16     <li>超实用的 CSS 代码段</li>
17     <li>超实用的 CSS 代码段</li>
18     <li>超实用的 CSS 代码段</li>
19     <li>超实用的 CSS 代码段</li>
20     <li>超实用的 CSS 代码段</li>
21     <li></li>
22     <li><a href="" class="buy_now">Buy Now</a></li>
23   </ul>
24   .....
25   <!-- 此处可根据需要添加结构代码 -->
26 </div>

```

//CSS 实现高亮部分代码

```

27 .pricing_table {
28   border:1px solid #c4cbcc;
29   border-radius:4px;
30   -moz-border-radius:4px;
31   -webkit-border-radius:4px;
32   outline:7px solid #f2f3f3;
33   float:left;
34 }
35 .pricing_table ul {
36   list-style:none;
37   float:left;
38   width:147px;
39   margin:0;
40   border:1px solid #f2f3f3;
41   padding:5px;
42   text-align:center;
43   background-color:#FFF;

```

```
44 }
45 .pricing_table ul:hover {
46     -webkit-transform: scale(1.1);
47     -moz-transform: scale(1.1);
48     -o-transform: scale(1.1);
49     -moz-box-shadow:3px 5px 7px rgba(0,0,0,.7);
50     -webkit-box-shadow: 3px 5px 7px rgba(0,0,0,.7);
51     box-shadow:3px 5px 7px rgba(0,0,0,.7);
52     cursor:pointer;
53 }
```

代码第 45 行，使用伪类: hover 设置当鼠标经过 ul 时为其设置样式，在第 46~48 行，使用 CSS 3 的 transform 属性，设置值为 scale 扩大 1.1 倍，加上-webkit-、-moz-、-o-前缀分别支持 Webkit、Mozilla、Opera 浏览器。注意 scale 接收一个参数，表示放大倍数。

实现效果如图 7.13 所示。当鼠标经过最中间一个元素时，该元素被放大而呈现的突出显示效果。由于使用 CSS 3 实现，因此该效果不可避免地对 IE 的支持有缺陷。



图 7.13 鼠标移动到 div 上高亮显示

7.11 发光锚链接

第 3.18 节介绍过锚链接的 4 种伪类: link、:visited、:hover、:active 的使用方法。本节介绍使用锚链接伪类实现锚链接的发光效果。实现代码如下：

```
//CSS 代码
01 a {
02     color: #00e;
03     font-size: 48px;
04 }
```

```

05 a:visited {
06     color: #551a8b;
07 }
08 a:hover {
09     color: #06e;
10 }
11 a:focus {
12     outline: thin dotted;
13 }
14 a:hover, a:active {
15     outline: 0;
16 }
17 a, a:visited, a:active {
18     text-decoration: none;
19     color: #fff;
20     -webkit-transition: all .3s ease-in-out;
21     -moz-transition: all .3s ease-in-out;
22     -o-transition: all .3s ease-in-out;
23 }
24 a:hover, .glow {
25     color: #ff0;
26     text-shadow: 0 0 10px #ff0;
27 }

```

代码第 1 行定义锚链接的颜色、字体，第 18 行去除锚链接的下画线，第 20 行使用 `transition` 属性设置锚链接发光的缓慢扩散的过程。

实现效果如图 7.14 所示。图中第 1 行文本是鼠标触发前的效果，第 2 行文本是鼠标触发锚链接后，锚链接显示的发光效果。



图 7.14 锚链接发光效果

7.12 屏蔽 Webkit 浏览器的高亮效果

访问移动网站时，在选中的元素周围会出现一些灰色的边框，可以通过 CSS 代码屏蔽这种样式。实现代码如下：

```
//CSS 代码
```



```

01 div {
02     padding: 20px;
03     background: #eee;
04 }
05 p {
06     -webkit-touch-callout: none; /* Chrome all / Safari all */
07     -webkit-user-select: none;      /* Chrome all / Safari all */
08     -khtml-user-select: none;      /* khtml all */
09     -moz-user-select: none;        /* Firefox all */
10     -ms-user-select: none;         /* IE 10+ */
11     user-select: none;
12 }
//HTML 代码
13 <div>
14     <p>在 WebKit 或 Firefox 浏览器无法选中进行复制。</p>
15 </div>
16
17 <div>
18     <p>在 WebKit 或 Firefox 浏览器无法选中</p>
19 </div>

```

在 iOS 设备中，在选中的元素上方，长按 3 秒后，iOS 会在选中的元素周围出现灰色边框并且弹出一个列表按钮，用户通过这些按钮可以在新窗口打开页面，而 a 标签的 target=“_self”指定当前窗口打开则失效了。代码第 6 行将 -webkit-touch-callout 属性设置为 none，来禁止 iOS 弹出列表按钮。第 3~7 行针对不同浏览器设置 user-select 的值为 none，禁止用户进行选择。实现效果如图 7.15 所示。

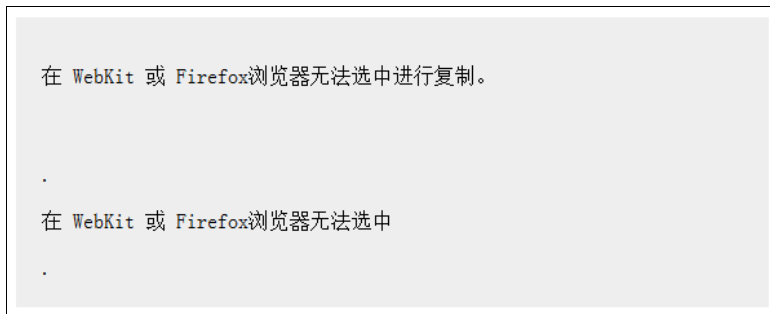


图 7.15 屏蔽 Webkit 移动浏览器中元素高亮效果

此外，屏蔽选择的样式 -webkit-user-select 属性有 3 个值。

- none: 子元素所有的文字都不能选择，包括 input 输入框中的文字也不能选择。
 - -moz-all: 子元素所有的文字都可被选择，但是 input 输入框中的文字不可以被选择。
 - -moz-none: 子元素所有的文字都不可被选择，但是 input 输入框中的文字除外。
- webkit-user-select 针对 Webkit 浏览器，在 IE 浏览器下是通过 onselectstart=“javascript: return false;” 事件来实现该功能。

7.13 多种风格的翻页页码

在综合性网站上，经常会涉及分页，根据网站设计风格的不同，所选择的翻页页码的风格也均有所不同。本节介绍几种不同风格的翻页页码。

7.13.1 Yahoo 旧版翻页风格

这里介绍 Yahoo 旧版实现的一种风格，实现代码如下：

```
//CSS 代码
01 div.yahoo{
02     padding-right:3px;
03     padding-left:3px;
04     padding-bottom:3px;
05     margin:3px;
06     padding-top:3px;
07     text-align:center
08 }
09 div.yahoo a{
10     border-right:#fff 1px solid;
11     padding-right:5px;
12     border-top:#fff 1px solid;
13     padding-left:5px;
14     padding-bottom:2px;
15     margin:2px;
16     border-left:#fff 1px solid;
17     color:#000099;
18     padding-top:2px;
19     border-bottom:#fff 1px solid;
20     text-decoration:underline
21 }
22 div.yahoo a:hover{
23     border-right:#000099 1px solid;
24     border-top:#000099 1px solid;
25     border-left:#000099 1px solid;
26     color:#000;
27     border-bottom:#000099 1px solid
28 }
29 div.yahoo a:active{
30     border-right:#000099 1px solid;
31     border-top:#000099 1px solid;
32     border-left:#000099 1px solid;
```

```

33     color:#f00;
34     border-bottom:#000099 1px solid
35 }
36 div.yahoo span.current{
37     border-right:#fff 1px solid;
38     padding-right:5px;
39     border-top:#fff 1px solid;
40     padding-left:5px;
41     font-weight:bold;
42     padding-bottom:2px;
43     margin:2px;
44     border-left:#fff 1px solid;
45     color:#000;
46     padding-top:2px;
47     border-bottom:#fff 1px solid;
48     background-color:#fff
49 }
50 div.yahoo span.disabled{
51     border:#eee 1px solid;
52     padding-right:5px;
53     padding-left:5px;
54     padding-bottom:2px;
55     margin:2px;
56     color:#ddd;
57     padding-top:2px;
58 }

```

代码第 1~8 行设置页码容器的样式，其中 `text-align` 设置为 `center`，即页码居中显示。

第 9~21 行为普通可点击页码的样式。第 22~28 行设置了鼠标移动到目标页码上方时该页码所呈现的样式，即有下画线，呈现蓝色边框，字体颜色为黑色。第 29~35 行设置了目标页码鼠标点击时，该页码显示的效果样式，呈现蓝色边框，字体颜色为红色。

第 36~49 行设计当前页码的样式，边框为白色，由于背景颜色也是白色，因此不显示当前页码的边框。同时，当前页码非链接，因此鼠标移到当前页码上不会有下画线效果。当前页码字体颜色为黑色、粗体。

第 50~58 行，设计了当“上一页”或“下一页”不可使用时的样式，即有灰色边框，颜色为暗灰色。使用 `padding` 与 `margin` 来设置内外间距。

实现效果如图 7.16 所示。

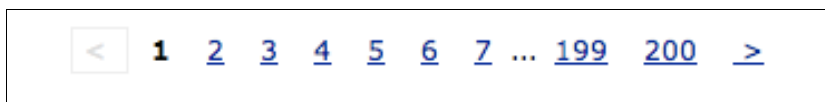


图 7.16 Yahoo 旧版翻页页码效果

7.13.2 Yahoo 新版翻页风格

新版的 Yahoo 翻页页码效果的实现代码如下：

//CSS 代码

```
01 .yahoo2{
02     padding-right:3px;
03     padding-left:3px;
04     font-size:0.85em;
05     padding-bottom:3px;
06     margin:3px;
07     padding-top:3px;
08     font-family:tahoma,helvetica,sans-serif;
09     text-align:center
10 }
11 div.yahoo2 a{
12     border-right:#ccdbe4 1px solid;
13     padding-right:8px;
14     background-position:50% bottom;
15     border-top:#ccdbe4 1px solid;
16     padding-left:8px;
17     padding-bottom:2px;
18     border-left:#ccdbe4 1px solid;
19     color:#0061de;
20     margin-right:3px;
21     padding-top:2px;
22     border-bottom:#ccdbe4 1px solid;
23     text-decoration:none
24 }
25 div.yahoo2 a:hover{
26     border-right:#2b55af 1px solid;
27     border-top:#2b55af 1px solid;
28     background-image:none;
29     border-left:#2b55af 1px solid;
30     color:#fff;
31     border-bottom:#2b55af 1px solid;
32     background-color:#3666d4
33 }
34 div.yahoo2 a:active{
35     border-right:#2b55af 1px solid;
36     border-top:#2b55af 1px solid;
37     background-image:none;
38     border-left:#2b55af 1px solid;
39     color:#fff;
40     border-bottom:#2b55af 1px solid;
```

```

41     background-color:#3666d4
42 }
43 div.yahoo2 span.current{
44     padding-right:6px;
45     padding-left:6px;
46     font-weight:bold;
47     padding-bottom:2px;
48     color:#000;
49     margin-right:3px;
50     padding-top:2px
51 }
52 div.yahoo2 span.disabled{
53     display:none
54 }
55 div.yahoo2 a.next{
56     border-right:#ccdbe4 2px solid;
57     border-top:#ccdbe4 2px solid;
58     margin:0px 0px 0px 10px;
59     border-left:#ccdbe4 2px solid;
60     border-bottom:#ccdbe4 2px solid
61 }
62 div.yahoo2 a.next:hover{
63     border-right:#2b55af 2px solid;
64     border-top:#2b55af 2px solid;
65     border-left:#2b55af 2px solid;
66     border-bottom:#2b55af 2px solid
67 }
68 div.yahoo2 a.prev{
69     border-right:#ccdbe4 2px solid;
70     border-top:#ccdbe4 2px solid;
71     margin:0px 10px 0px 0px;
72     border-left:#ccdbe4 2px solid;
73     border-bottom:#ccdbe4 2px solid
74 }
75 div.yahoo2 a.prev:hover{
76     border-right:#2b55af 2px solid;
77     border-top:#2b55af 2px solid;
78     border-left:#2b55af 2px solid;
79     border-bottom:#2b55af 2px solid
80 }

```

代码第 1~8 行设置页码容器的样式，其中 `text-align` 设置为 `center`，即页码居中显示。第 11~24 行为普通可点击页码的样式。边框为 1 像素实线，边框颜色十六进制值为 `#ccdbe4`，即浅蓝色。字体颜色为蓝色。设置内、外间距，以使页码数字居中呈现在方框内部。

第 25~33 行设置了鼠标移动到目标页码上方时该页码所呈现的样式，即无下划线，呈

现蓝色背景，字体颜色为白色。第 34~42 行设置了目标页码鼠标点击时，该页码显示的效果样式，呈现蓝色背景，字体颜色为白色。第 43~51 行设计当前页码的样式，其样式与 Yahoo 旧版样式基本一致。第 52~54 行，设计不可使用页码的样式，即当“上一页”或“下一页”不可使用时则不显示。

实现效果如图 7.17 所示。



图 7.17 Yahoo 新版翻页页码效果

7.13.3 Meneame 翻页风格

Meneame 与 Yahoo 新版的翻页效果类似，有区别的是，设计以黄色为主色调，去除了下一页的英文字母，仅留下右箭头。另外，显示“上一页”与“下一页”可点击与不可点击状态。实现代码如下：

```
//CSS 代码
01 div.meneame{
02     padding-right:3px;
03     padding-left:3px;
04     font-size:80%;
05     padding-bottom:3px;
06     margin:3px;
07     color:#ff6500;
08     padding-top:3px;
09     text-align:center
10 }
11 div.meneame a{
12     border-right:#ff9600 1px solid;
13     padding-right:7px;
14     background-position:50% bottom;
15     border-top:#ff9600 1px solid;
16     padding-left:7px;
17     background-image:url(meneame.jpg);
18     padding-bottom:5px;
19     border-left:#ff9600 1px solid;
20     color:#ff6500;
21     margin-right:3px;
22     padding-top:5px;
23     border-bottom:#ff9600 1px solid;
24     text-decoration:none
25 }
26 div.meneame a:hover{
```

```

27     border-right:#ff9600 1px solid;
28     border-top:#ff9600 1px solid;
29     background-image:none;
30     border-left:#ff9600 1px solid;
31     color:#ff6500;
32     border-bottom:#ff9600 1px solid;
33     background-color:#ffc794
34 }
35 div.meneame a:active{
36     border-right:#ff9600 1px solid;
37     border-top:#ff9600 1px solid;
38     background-image:none;
39     border-left:#ff9600 1px solid;
40     color:#ff6500;
41     border-bottom:#ff9600 1px solid;
42     background-color:#ffc794
43 }
44 div.meneame span.current{
45     border-right:#ff6500 1px solid;
46     padding-right:7px;
47     border-top:#ff6500 1px solid;
48     padding-left:7px;
49     font-weight:bold;
50     padding-bottom:5px;
51     border-left:#ff6500 1px solid;
52     color:#ff6500;
53     margin-right:3px;
54     padding-top:5px;
55     border-bottom:#ff6500 1px solid;
56     background-color:#ffbe94
57 }
58 div.meneame span.disabled{
59     border-right:#ffe3c6 1px solid;
60     padding-right:7px;
61     border-top:#ffe3c6 1px solid;
62     padding-left:7px;
63     padding-bottom:5px;
64     border-left:#ffe3c6 1px solid;
65     color:#ffe3c6;
66     margin-right:3px;
67     padding-top:5px;
68     border-bottom:#ffe3c6 1px solid
69 }

```

Meneame 与 Yahoo 新版的翻页效果的实现代码类似,有区别的地方是在第 44~57 行设

置了当前页码样式，仍然呈现边框，且呈现橘黄色背景色，以突出呈现当前页码。并且，在第 58~69 行，设置了“上一页”与“下一页”可点击与不可点击状态。

实现效果如图 7.18 所示。



图 7.18 Menename 翻页页码效果

7.13.4 YouTube 翻页风格

YouTube 的翻页页码带有灰色底纹，且整体页码居右显示。实现代码如下：

```
//CSS 代码
01 div.youtube{
02     padding-right:6px;
03     border-top:#9c9a9c 1px dotted;
04     padding-left:0px;
05     font-size:13px;
06     padding-bottom:4px;
07     color:#313031;
08     padding-top:4px;
09     font-family:arial,Helvetica,sans-serif;
10     background-color:#cecfce;
11     text-align:right;
12 }
13 div.youtube a{
14     padding-right:3px;
15     padding-left:3px;
16     font-weight:bold;
17     padding-bottom:1px;
18     margin:0px 1px;
19     color:#0030ce;
20     padding-top:1px;
21     text-decoration:underline;
22 }
23 div.youtube span.current{
24     padding-right:2px;
25     padding-left:2px;
26     padding-bottom:1px;
27     color:#000;
28     padding-top:1px;
29     background-color:#fff
30 }
31 div.youtube span.disabled{
```



```

32     display:none
33 }

```

代码第 11 行, 设置 `text-align` 为 `right`, 因此, 页码布局整体居右显示。第 27 行设置前景色为黑色, 第 29 行设置了当前页码的背景色为白色。

实现效果如图 7.19 所示。



图 7.19 YouTube 翻页页码效果

7.14 创建针线缝合效果

什么是针线缝合效果? 针线缝合效果就是在元素的边框上可以看到仿针线的缝制效果。在一些网站上可以看到针线缝合效果。本节将对针线缝合效果的实现做一个详细的介绍。针线缝合效果的实现代码如下:

```

//CSS 代码
01 .noshadow_div_solid_stitch{
02     height:100px;
03     width:500px;
04     padding:20px;
05     margin:15px;
06     background:#6E6E6E;
07     border:2px solid rgba(247, 231, 247, 0.8);
08     -moz-border-radius:6px;
09     -webkit-border-radius:6px;
10     border-radius:6px;
11     font-size:24px;
12     color: white;
13     position: relative;
14     z-index:0;
15 }
16 .noshadow_div_solid_stitch:before{
17     border: 2px dashed rgba(255,255,255,0.8);
18     top: 3px;
19     bottom: 3px;
20     left: 3px;
21     right: 3px;
22     position: absolute;
23     z-index:-1;
24     content:"";
25 }

```

```
//HTML 代码
```

```
<div class="noshadow_div_solid_stitch">超实用的 CSS 代码段 — 针线缝合效果</div>
```

针线缝合效果的实现使用了 CSS 3 的伪元素:before、属性 border-radius。代码第 7 行定义了 border 属性,设置目标元素边框为 2 像素宽、呈虚线效果及边框的颜色值。第 10 行设置圆角边框,圆角半径为 6 像素。第 16 行使用:before 伪元素设置边框距离元素实际上、下、左、右距离均为 3 像素,使其呈现出一个间隙,并最终实现一个针线缝合的整体效果。

针线缝合的效果如图 7.20 所示。

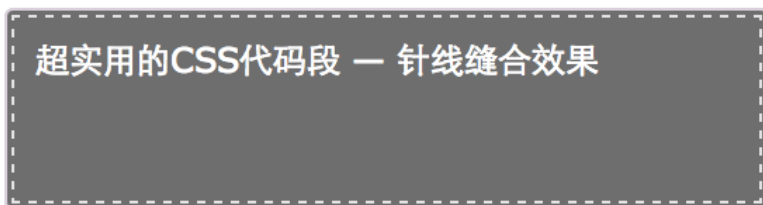


图 7.20 针线缝合效果

第 8 章 盒子

网页设计中的每个元素都是长方形的盒子。盒子模型规定了元素盒子的宽度、高度、内边距、边框和外边距的显示方式。本章将重点讲述如何使用 CSS 3 盒子模型、内/外层盒阴影，以及利用 CSS 3 盒模型的特点制作水晶盒等特效。

本章主要涉及的内容有：

- 内层 CSS 3 盒模型
- 外层 CSS 3 盒阴影
- 纯 CSS 3 透明水晶盒
- 投影发光效果

8.1 CSS 3 盒模型

在网页设计中，每个 HTML 标记都可看作一个盒子，每个盒子都有外边界 `margin`、边框 `border`、填充 `padding`、内容 `content` 等 4 个属性，每个属性又可分别在上、右、下、左 4 个方向上设置属性，也可同时设置属性。如有以下代码分别设置元素的盒模型属性：

```
//CSS 代码
01 #mydiv {
02     margin-top: 10px;
03     margin-right: 11px;
04     margin-bottom: 12px;
05     margin-left: 13px;
06     padding-right: 10px;
07     padding-left: 20px;
08     padding-top: 5px;
09     padding-bottom: 15px;
10     border-top-width: 1px;
11     border-left-width: 4px;
12     border-right-width: 2px;
13     border-bottom-width: 3px;
14 }
```

以上代码所设置的元素的盒模型如 8.1 所示。对 4 个方向上的外边界 `margin`、边框 `border`、填充 `padding`、内容 `content` 等 4 个属性分别设置了不同的值。

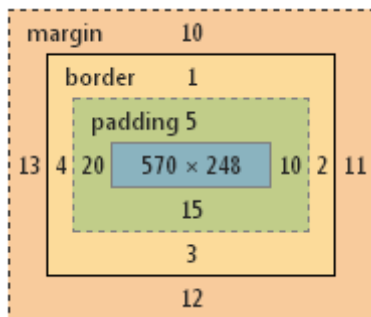


图 8.1 CSS 盒子模型

CSS 3 引入了 `box-flex` 属性，可定义弹性盒子模型。例如，通过以下代码可实现 CSS 3 的盒子模型布局：

```
//CSS 代码
01 container {
02     width: 1000px;
03     display: -webkit-box;
04     display: -moz-box;
05     -webkit-box-orient: horizontal;
06     -moz-box-orient: horizontal;
07 }
08 .blue {
09     background: #357c96;
10     font-weight: bold;
11     margin: 2px;
12     padding: 20px;
13     color: #fff;
14     font-family: Arial, sans-serif;
15 }
//HTML 代码
16 <div class="container">
17   <div class="blue"> 超实用 </div>
18   <div class="blue"> CSS 代码段 </div>
19   <div class="blue"> 超实用的 CSS 代码段 </div>
20   <div class="blue">超实用的 CSS 代码段!</div>
21 </div>
```

代码第 3~4 行为 `.container` 容器定义了 `display` 属性的值为盒子 `-webkit-box/ -moz-box`，在第 5~6 行设置了盒子方向为水平方向，其展示效果如图 8.2 所示。可以看出，使用 CSS 3 可通过简单的代码便捷地实现盒子布局，实现了过去只能通过定位或浮动才能实现的效果。

另外，也可以设置 `box-orient` 属性值为 `vertical`（即垂直方向）。



图 8.2 CSS 3 弹性盒子模型

8.2 内层 CSS 3 盒阴影

盒阴影 (box shadow) 属性是 CSS 3 的另一个盒模型属性, 几乎可以运用在任何元素上, 并且可设置内层盒阴影与外层盒阴影。盒阴影通过 CSS3 的 box-shadow 属性实现, 语法如下:

```
01 box-shadow: h-shadow v-shadow blur spread color inset;
```

其中, h-shadow 为阴影水平方向偏移, v-shadow 为阴影垂直方向偏移, blur 为阴影模糊羽化距离, color 为阴影的颜色, spread 为阴影尺寸, inset 将外部阴影设置为内部阴影。

下面这段代码可实现内层阴影:

```
//CSS 代码
01 #mydiv {
02     width: 500px;
03     height: 200px;
04     margin-left: 50px;
05     -webkit-box-shadow: inset 2px 3px 4px 10px #000;
06     -moz-box-shadow: inset 2px 3px 4px 10px #000;
07     -o-box-shadow: inset 2px 3px 4px 10px #000;
08     box-shadow: inset 2px 3px 4px 10px #000;
09 }
//HTML 代码
10 <div id="mydiv"> 超实用的 CSS 代码段 超实用的 CSS 代码段 </div>
```

第 5~8 为不同的浏览器设置了盒阴影的兼容方案, 设置内部阴影水平偏移 2px、垂直偏移 3px、羽化 4px、阴影尺寸 10px、颜色为黑色。实例效果如图 8.3 所示。



图 8.3 内层 CSS 3 盒阴影

说明: IE9+、Firefox 4、Chrome、Opera 以及 Safari 5.1.1 均支持 box-shadow 属性。

8.3 外层 CSS 3 盒阴影

下面介绍一段外层阴影代码设计:

```
//CSS 代码
01 #mydiv {
02     width: 300px;
```

```

03     height:100px;
04     color: #fff;
05     padding: 20px;
06     background-color:#ff9900;
07     -webkit-box-shadow: 10px 10px 5px #888;
08     -moz-box-shadow: 10px 10px 5px #888;
09     -o-box-shadow: 10px 10px 5px #888;
10     box-shadow: 10px 10px 5px #888888;
11 }
//HTML 代码
12 <div id="mydiv"> 超实用的 CSS 代码段 超实用的 CSS 代码段 </div>

```

在代码第 7~10 行设置了盒子 mydiv 的外阴影，其水平与垂直偏移均为 10px，颜色值为 #888。实现效果如图 8.4 所示。



图 8.4 外层 CSS 3 盒阴影

8.4 纯 CSS 3 透明水晶盒

第 8.1 节介绍了在 CSS 3 中为元素设置弹性盒子布局。本节介绍如何不使用 JavaScript、SVG 等其他技术，仅使用 CSS 技术实现透明水晶盒。通过设置 CSS 3 的 transform 属性的 skew、rotate 等参数，将元素组合成一个具有 6 个面的 3D 对象。

首先，进行 HTML 结构设计，水晶盒的每一个面为一个独立的 div 元素。代码如下：

```

01 <div class="box under">
02     <div class="inBox box_left">超实用的 CSS 代码段</div>
03     <div class="inBox box_back">超实用的 CSS 代码段</div>
04     <div class="inBox box_bottom">超实用的 CSS 代码段</div>
05     <div class="inBox box_right">
06         <div class="conBox">
07             <h3 class="title">超实用的 CSS 代码段</h3>
08             <p>超实用的 CSS 代码段...超实用的 CSS 代码段...</p>
09         </div>
10     </div>
11     <div class="inBox box_before">
12         <div class="conBox">
13             <h3 class="title">超实用的 CSS 代码段</h3>
14             <p>超实用的 CSS 代码段...超实用的 CSS 代码段...</p>

```

```

15         </div>
16     </div>
17     <div class="inBox box_top"><br><br>超实用的 CSS 代码段</div>
18 </div>

```

使用一个 div 包含“前、后、左、右、上、下”6个面，因为定位产生层高的关系，所以书写的顺序是“后、下、左、前、上、右”，由于较后定义的 div 会顺序层叠在较早定义的 div 元素之上，这样可以不再另外设定 z-index 属性定义层叠次序。

其次，进行 CSS 设计。前、后、左、右4个面的难度系数相同，使用 skew 就可以实现倾斜效果，然后就是再分别为各个面进行定位。前、后两个面用 skew(0deg,20deg)，即 Y 轴正向倾斜，左、右两个面用 skew(0deg,-20deg)在 Y 轴负向倾斜，然后再定位对齐，基本的框就出来了。具体的实现代码如下：

```

01 *{
02     margin: 0px;                /*去除元素默认内外边框*/
03     padding: 0px;
04 }
05 body{
06     font-size: 12px;
07     background: #ccc;           /*设置全局字体大小与背景颜色*/
08 }
09 .box{
10     position: relative;         /*设置水晶盒容器为相对定位*/
11     left: 500px;
12     top: 300px;
13 }
14 .box .inBox{
15     height: 200px;              /*设置水晶盒6个面的样式*/
16     width: 200px;              /*在此设置的宽高，后面代码会根据面做调整*/
17     background: rgba(255,255,255,0.2);
18     position: absolute;         /*设置水晶盒6个面均绝对定位*/
19     -webkit-transform-origin: 0% 0%;
20 }
21 /*设置伪类:after 与:before 为绝对定位*/
22 .box .inBox:after,.under .box_before:before,.under .box_right:before{
23     background: rgba(0,0,0,0);
24     display: block;
25     height: 200px;
26     width: 200px;
27     content: " ";
28     position: absolute;
29     left: 0px;
30     top: 0px;
31 }
32 .box .box_before{               /*设置水晶盒前面正向倾斜 20°*/

```

```

33     -webkit-transform: skew(0deg,20deg);
34 }
35 .box .box_back {                                /*设置水晶盒后面正向倾斜 20°, 并设置定位*/
36     -webkit-transform: skew(0deg,20deg);
37     left: 200px;
38     top: -73px;
39 }
40 .box .box_left {                                /*设置水晶盒左面负向倾斜 20° */
41     -webkit-transform: skew(0deg,-20deg);
42     background: rgba(255,255,255,0.1);
43 }
44 .box .box_right {                               /*设置水晶盒右面负向倾斜 20°, 并设置定位*/
45     -webkit-transform: skew(0deg,-20deg);
46     left: 200px;
47     top: 73px;
48     background: rgba(255,255,255,0.2);
49 }
50 .box .box_top {                                 /*设置水晶盒上面先旋转再倾斜*/
51     -webkit-transform: rotate(-40deg) skew(30deg,20deg);
52     height: 185px;
53     background: rgba(255,255,255,0.4);
54 }
55 .box .box_bottom {                             /*设置水晶盒下面先旋转再倾斜*/
56     -webkit-transform: rotate(-40deg) skew(30deg,20deg);
57     height: 185px;
58     top: 200px;
59     background: rgba(255,255,255,0.3);
60 }
61 /*通过伪类:after 设置水晶盒透明度*/
62 .box .box_bottom:after {
63     height: 185px;
64     -webkit-box-shadow: 0px -1px 0px rgba(255,255,255,0.4),1px 0px 0px rgba(255,
        255,255,0.4),-1px 1px 5px rgba(50,50,50,0.1);
65 }
66 .box .box_before:after {
67     -webkit-box-shadow: 0px -1px 0px rgba(255,255,255,0.6),1px 0px 0px rgba(255,
        255,255,0.6),0px 1px 0px rgba(255,255,255,0.6),-1px 0px 0px rgba(255,255,255,0.6);
68     background: -webkit-linear-gradient(-45deg,rgba(255,255,255,0.3),rgba(255,255,255,0.2)
        40%,rgba(255,255,255,0.2) 70%,rgba(255,255,255,0.1));
69 }
70 .box .box_right:after {
71     -webkit-box-shadow: 0px -1px 0px rgba(255,255,255,0.6),1px 0px 0px rgba(255,
        255,255,0.6),0px 1px 0px rgba(255,255,255,0.6);
72     background: -webkit-linear-gradient(-45deg,rgba(255,255,255,0.2),rgba(255,255,255,0.1)
        40%,rgba(255,255,255,0.1) 40%,rgba(255,255,255,0));

```



```

73 }
74 .box .box_top:after {
75     height: 185px;
76     -webkit-box-shadow: 0px -1px 0px rgba(255,255,255,0.6),1px 0px 0px rgba(255,
255,255,0.6);
77     background: -webkit-linear-gradient(-45deg,rgba(255,255,255,0.2),rgba(255,255,255,0.1)
40%,rgba(255,255,255,0.1) 40%,rgba(255,255,255,0));
78 }
79 .box .box_left:after {
80     -webkit-box-shadow: 1px 0px 0px rgba(255,255,255,0.2);
81     background: -webkit-linear-gradient(-65deg,rgba(255,255,255,0.1),rgba(255,255,255,0)
50%,rgba(255,255,255,0.1));
82 }
83 .box .box_back:after {
84     background: -webkit-linear-gradient(top,rgba(255,255,255,0),rgba(255,255,255,0.1),
rgba(255,255,255,0));
85 }
86 /*通过伪类:before 设置水晶盒边框，不影响水晶盒原有宽高设计*/
87 .under .box_before:before,.under .box_right:before {
88     -webkit-transform: translate(0px,201px);
89     background: -webkit-linear-gradient(top,rgba(255,255,255,0.2),rgba(255,255,255,0.3) 2%,
rgba(255,255,255,0) 50%,rgba(255,255,255,0));
90 }
91 .under .box_before:before {
92     width: 201px;
93 }
94 /*设置水晶盒的前、右、上3个面鼠标经过的旋转倾斜效果*/
95 .box .box_before:hover {
96     -webkit-transform: skew(0deg,0deg) translate(0px,73px);
97 }
98 .box .box_right:hover {
99     -webkit-transform: skew(0deg,0deg);
100 }
101 .box .box_top:hover {
102     -webkit-transform: rotate(-80deg) skew(30deg,20deg);
103 }
104 /*重设水晶盒前、后两个面的宽度，形成长方形*/
105 .box .box_before,.box .box_back,.under .box_before:before {
106     width: 300px;
107 }
108 .box .box_right {
109     left: 300px;
110     top: 110px;
111 }
112 /*重设水晶盒底面的高度，形成长方形*/

```

```

113 .box .box_top,.box .box_bottom {
114     height: 278px;
115 }
116 /*重设水晶盒前、后两个面伪类的高度，以设置长方体在面上的背景透明*/
117 .box .box_bottom:after {
118     height: 280px;
119 }
120 .under .box_before:after {
121     width: 300px;
122 }

```

第 10 行，将水晶盒的外部包裹容器设置为相对定位。第 28 行，将水晶盒的 6 个面都设置为相对于外部包裹容器绝对定位。前、后、上、下 4 个面的尺寸是 300*200px，左、右 2 个面的尺寸是 200*200px。

第 32、35 行设置前、后两面 `skew(0deg,20deg)`，即在 Y 轴倾斜 20°。第 40、44 行设置左、右两面 `skew(0deg, -20deg)`，即在 Y 轴负向倾斜 20°。上、下两个面稍微复杂一些，得进行旋转再倾斜。如果先倾斜再选择效果会与预期不一致，这是因为确定了的倾斜效果会因为后面的旋转而变形。第 50、55 行分别设置了上、下两个面的旋转倾斜效果。

为了在设置边框时不影响水晶盒原有设计的长、宽、高，使用在各个面的 `:after` 伪类上，对各个边进行设置 `box-shadow` 属性。在背景上，把上、前、右对着浏览者的透明度调低于显示在后面的各个面，将半透明到透明的渐变叠加到基本的盒子颜色上，产生层次。第 66、70、74、79、83、87 行代码实现了这种设置，由左上到右下拉斜线渐变，从 0% 0.3 到 40% 0.2，70% 0.2 到 100% 0.1。

使用各个面的伪类 `:before` 设置投影效果，且伪类层是相对于原图层的，所以修改原图层位置变形状态的时候，伪类层也会相应的修改。面向用户的只有前、右两个面，因此做这两个投影就够了。代码第 87~90 行实现了这一设计，`translate(0px, 201px)` 实现了在 Y 轴上移动 200px，从第 201px 上开始显示，并设置背景用 `linear-gradient` 实现渐变。

该 CSS 3 透明长方体水晶盒的实现效果如图 8.5 所示。

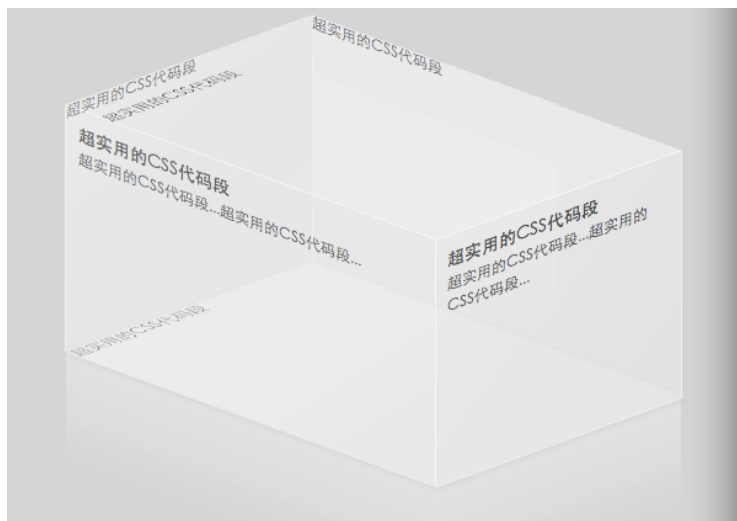


图 8.5 CSS 3 透明水晶盒

注意：目前本例代码仅支持基于 WebKit、Gecko 的浏览器。

8.5 投影发光效果

在第 8.3 和第 8.4 节中，介绍了使用 CSS 3 中的 box-shadow 实现盒投影。本节介绍通过使用 text-shadow 实现文字投影，其产生的阴影将随着文字轮廓进行投影。在设计的过程中，设计背景颜色深、投影颜色较浅，则呈现出发光效果，反之，则为投影效果。图 8.6 与图 8.7 分别为文字阴影与盒子阴影的对比图，文字阴影实现代码为 text-shadow:2px 2px 4px #000，盒子阴影实现代码为 box-shadow:2px 2px 4px #000。其中盒子阴影会根据存放文字的盒子容器进行投影。



图 8.6 文字阴影



图 8.7 盒子阴影

由于 text-shadow 没有 inset 属性，为了实现文字内发光、渐变效果，可以采用蒙版、:before、:after 层叠的方式来实现。代码如下：

```
//CSS 代码
01 body{
02     background:#CCC;
03 }
04 .text {
05     font-size: 100px;
06     font-family: "微软雅黑";
07     font-weight: bold;
08     position: relative;
09     text-shadow: 1px 2px 3px rgba(67,8,7,0.8);
10     color: #c60000;
11 }
12 .text:after {
13     position: absolute;
14     left: 0px;
15     content: "超实用的 CSS 代码段";
16     color: #ea0000;
17     -webkit-mask-image: -webkit-gradient(linear, 0 0, 0 100%, from(rgba(0, 0, 0, 0.9)),
        color-stop(40%, rgba(0, 0, 0, 0.5)),to(rgba(0, 0, 0, 0)));
```

```
18     text-shadow: 0px 0px 2px rgba(234,0,0,1);
19 }
20 .text:before {
21     content: "超实用的 CSS 代码段";
22     text-shadow: 0px 0px 5px rgba(110,0,0,0.8);
23     position: absolute;
24     left: 0px;
25     color: rgba(0,0,0,0);
26     -webkit-mask-image: -webkit-gradient(linear, 0 0, 0 100%, from(rgba(0, 0, 0, 0)),
    color-stop(50%, rgba(0, 0, 0, 0.4)), color-stop(70%, rgba(0, 0, 0, 0.7)),to(rgba(0, 0, 0, 0.9)));
27     z-index: 99;
28 }
29 .box {
30     margin-top: 100px;
31     box-shadow:2px 2px 4px #000;
32     width:400px;
33     font-size:40px;
34     height:40px;
35     color:#000;
36     padding:5px;
37 }
//HTML 代码
38 <div class="text">超实用的 CSS 代码段</div>
39 <div class="box">超实用的 CSS 代码段</div>
```

代码第 4~11 行为 .text 元素设置底色；第 12~19 行为 .text:after 添加 -webkit-mask-image 蒙版属性，实现渐变；第 20~28 行为 .text:before 设置 -webkit-mask-image，其渐变色更为突出，以提升渐变效果。 .text:after 与 .text:before 属性均为绝对定位，定位于 .text 元素的上方，层叠以实现内发光效果。本例效果如图 8.8 所示。

超实用的CSS代码段

图 8.8 文字发光阴影效果

第 9 章 3D 相关 ➞

随着 CSS 3 的流行，过去只能通过图片、Flash、JavaScript 等实现的效果，慢慢都用 CSS 技术来实现了。尤其是与 3D 相关的一些特效，CSS 3 实现的方法更为简单、网页性能更为高效。

本章主要涉及的内容有：

- 3D 文字
- 3D 图片立体的设计与实现
- 3D 按钮、3D 下拉菜单
- 3D 旋转动画

9.1 3D 文字

本节介绍如何通过堆叠多层阴影来创建 3D 文字，并进一步利用 CSS 3 的 `transform` 和 `transition` 属性来实现鼠标移过字体放大的效果。为了实现 3D 的文字效果，我们将会利用 CSS 3 的 `text-shadow` 属性，`text-shadow` 的工作原理如下：

```
//CSS 代码
01 .text-class{
02     text-shadow: [X offset] [Y offset] [Blur size] [Colour];
03 }
```

注意：X 表示 *x* 轴上的位移，可为负值；Y 表示 *y* 轴上的位移，可为负值；Blur 表示投影的宽度，不能为负值；Color 为投影的颜色。

下面举例说明使用 `text-shadow` 实现 3D 的文字效果，代码如下：

```
//CSS 代码
01 #demo1 {
02     padding: 100px;
03     background-color: #F0EFE5;
04 }
05 #demo2 {
06     padding: 10%;
07     background: #6280b7;
08 }
```

```

09 h2{
10     font-size: 80px;
11     color: #fff;
12     letter-spacing: 5px;
13     text-shadow:                /* 设置 text-shadow 属性，多层阴影堆叠*/
14     1px 1px 0 #CCC,
15     2px 2px 0 #CCC,
16     /* end of 2 level deep grey shadow */
17     3px 3px 0 #444,
18     4px 4px 0 #444,
19     5px 5px 0 #444,
20     6px 6px 0 #444;
21     /* end of 4 level deep dark shadow */
22     /* CSS 3 Transition Effect */
23     -webkit-transition: all 0.12s ease-out;    /* Safari & Chrome */
24     -moz-transition: all 0.12s ease-out;      /* Firefox */
25     -o-transition: all 0.12s ease-out;       /* Opera */
26 }
27 h2:hover
28 {
29     /* 设置 CSS 3 Transform 的放大效果*/
30     -webkit-transform: scale(1.2);            /* Safari & Chrome */
31     -moz-transform: scale(1.2);              /* Firefox */
32     -o-transform: scale(1.2);                /* Opera */
33 }
34 p{
35     letter-spacing: 4px;
36     text-shadow:                /* 设置 text-shadow 属性 */
37     1px 0px #546a92,            /* 通过设置 X 与 Y 的不同偏移量来实现多层阴影堆叠 */
38     0px 1px #384762,
39     2px 1px #546a92,
40     1px 2px #384762,
41     3px 2px #546a92,
42     2px 3px #384762,
43     4px 3px #546a92,
44     3px 4px #384762,
45     5px 4px #546a92,
46     4px 5px #384762,
47     6px 5px #546a92,
48     5px 6px #384762,
49     7px 6px #546a92,
50     6px 7px #384762,
51     8px 7px #546a92,
52     7px 8px #384762,
53     8px 8px #546a92;

```

```

54 }
//HTML 代码
55 <div id="demo1">
56     <h2>超实用的 CSS 代码段</h2>
57 </div>
58 <div id="demo2">
59     <p>超实用的 CSS 代码段</p>
60 </div>

```

此段代码是通过堆叠多层投影实现 3D 效果。代码第 14~20 行是为#demo1 中文字 3D 效果的投影堆叠实现过程。文字颜色为白色，深灰色为投影颜色。第 30~32 行使用 transform 的 scale(1.2)来实现文字放大效果，并在第 23~25 行设置放大的淡入淡出效果。#demo2 的 3D 文字使用更多的投影堆叠，其实现的 3D 效果更加明显，实现代码是第 34~54 行，这是使用 text-shadow 的 3D 文字。

本例效果如图 9.1 所示。



图 9.1 3D 文字

注意：此 3D 文字效果是用纯 CSS 写的，没有使用 JavaScript，并且需要用支持 CSS 3 的浏览器才能看出效果，如 Firefox、Chrome、Safari 和 Opera。

9.2 3D 图片立体效果

为了实现 3D 的图片立体效果，我们将会利用 CSS 3 的 transform-style 和 perspective 属性。

transform-style 属性规定如何在 3D 空间中呈现被嵌套的元素，其可选值有 flat 或 preserve-3d，当值为 flat 时，其子元素将不保留其 3D 位置；当值为 preserve-3d 时，其子元素将保留其 3D 位置。

perspective 属性为一个元素设置三维透视的距离，仅作用于元素的后代，而不是该元素本身。当 perspective: none/0; 时，相当于没有设置 perspective 属性。例如当建立一个立方体时，长、宽、高均为 200px，若设置 perspective < 200px，相当于站在盒子里面看到的

效果，若设置 `perspective > 200px`，相当于站在远离立方体位置看到的效果。当元素没有设置 `perspective` 时，所有后代元素在同一个二维平面上，不存在景深的效果；设置 `perspective` 后，可看到三维的效果。

3D 图片的立体效果正是利用了以上这两个属性来设置景深的效果。本例代码如下：

```
//CSS 代码
01 body {                                /* 设置背景颜色 */
02     background: #ddd;
03 }
04 .view {
05     width: 980px;
06     height: 400px;
07     margin-top: 20px;
08     float: left;
09     position: relative;
10     border: 8px solid #fff;
11     box-shadow: 1px 1px 2px rgba(0,0,0,0.05);
12     background: #333;
13     /* 设置 perspective 属性可实现景深 */
14     -webkit-perspective: 1000px;
15     -moz-perspective: 1000px;
16     -o-perspective: 1000px;
17     -ms-perspective: 1000px;
18     perspective: 1000px;
19 }
20 .view .slice{
21     width: 196px;
22     height: 100%;
23     z-index: 100;
24     /* 设置 preserve-3d */
25     -webkit-transform-style: preserve-3d;
26     -moz-transform-style: preserve-3d;
27     -o-transform-style: preserve-3d;
28     -ms-transform-style: preserve-3d;
29     transform-style: preserve-3d;
30     /* 设置旋转元素的基点位置 */
31     -webkit-transform-origin: left center;
32     -moz-transform-origin: left center;
33     -o-transform-origin: left center;
34     -ms-transform-origin: left center;
35     transform-origin: left center;
36     /* 设置变化效果 */
37     -webkit-transition: -webkit-transform 150ms ease-in-out;
38     -moz-transition: -moz-transform 150ms ease-in-out;
39     -o-transition: -o-transform 150ms ease-in-out;
```



```

40     -ms-transition: -ms-transform 150ms ease-in-out;
41     transition: transform 150ms ease-in-out;
42
43 }
44 /* 利用 translate3d 属性设置 x/y/z 这 3 个维度的移动 */
45 .view .s2,
46 .view .s3,
47 .view .s4,
48 .view .s5 {
49     -webkit-transform: translate3d(196px,0,0);
50     -moz-transform: translate3d(196px,0,0);
51     -o-transform: translate3d(196px,0,0);
52     -ms-transform: translate3d(196px,0,0);
53     transform: translate3d(196px,0,0);
54 }
55 /* 设置背景图片的位置 */
56 .view .s1 {
57     background-position: 0px 0px;
58 }
59 .view .s2 {
60     background-position: -196px 0px;
61 }
62 .view .s3 {
63     background-position: -392px 0px;
64 }
65 .view .s4 {
66     background-position: -588px 0px;
67 }
68 .view .s5 {
69     background-position: -784px 0px;
70 }
71 .view .overlay {
72     width: 196px;
73     height: 100%;
74     opacity: 0;           /* 默认情况下不呈现 3D 图 */
75     position: absolute;
76     -webkit-transition: opacity 150ms ease-in-out;
77     -moz-transition: opacity 150ms ease-in-out;
78     -o-transition: opacity 150ms ease-in-out;
79     -ms-transition: opacity 150ms ease-in-out;
80     transition: opacity 150ms ease-in-out;
81 }
82 .view:hover .overlay {
83     opacity: 1;           /* 当鼠标移动到图片上时呈现 3D 图效果 */
84 }

```

```

85 .view img {
86     position: absolute;
87     z-index: 0;
88     -webkit-transition: left 0.3s ease-in-out;
89     -o-transition: left 0.3s ease-in-out;
90     -moz-transition: left 0.3s ease-in-out;
91     -ms-transition: left 0.3s ease-in-out;
92     transition: left 0.3s ease-in-out;
93 }
94 /* 为 s2/s3/s4/s5 分别设置 3D 移动与旋转效果 */
95 .view:hover .s2{
96     -webkit-transform: translate3d(195px,0,0) rotate3d(0,1,0,-45deg);
97     -moz-transform: translate3d(195px,0,0) rotate3d(0,1,0,-45deg);
98     -o-transform: translate3d(195px,0,0) rotate3d(0,1,0,-45deg);
99     -ms-transform: translate3d(195px,0,0) rotate3d(0,1,0,-45deg);
100     transform: translate3d(195px,0,0) rotate3d(0,1,0,-45deg);
101 }
102 .view:hover .s3,
103 .view:hover .s5{
104     -webkit-transform: translate3d(195px,0,0) rotate3d(0,1,0,90deg);
105     -moz-transform: translate3d(195px,0,0) rotate3d(0,1,0,90deg);
106     -o-transform: translate3d(195px,0,0) rotate3d(0,1,0,90deg);
107     -ms-transform: translate3d(195px,0,0) rotate3d(0,1,0,90deg);
108     transform: translate3d(195px,0,0) rotate3d(0,1,0,90deg);
109 }
110 .view:hover .s4{
111     -webkit-transform: translate3d(195px,0,0) rotate3d(0,1,0,-90deg);
112     -moz-transform: translate3d(195px,0,0) rotate3d(0,1,0,-90deg);
113     -o-transform: translate3d(195px,0,0) rotate3d(0,1,0,-90deg);
114     -ms-transform: translate3d(195px,0,0) rotate3d(0,1,0,-90deg);
115     transform: translate3d(195px,0,0) rotate3d(0,1,0,-90deg);
116 }
117 /* 使用 linear-gradient 设置背景渐变 */
118 .view .s1 > .overlay {
119     background: -moz-linear-gradient(right, rgba(0,0,0,0.05) 0%, rgba(0,0,0,0) 100%);
120     background: -webkit-linear-gradient(right, rgba(0,0,0,0.05) 0%,rgba(0,0,0,0) 100%);
121     background: -o-linear-gradient(right, rgba(0,0,0,0.05) 0%,rgba(0,0,0,0) 100%);
122     background: -ms-linear-gradient(right, rgba(0,0,0,0.05) 0%,rgba(0,0,0,0) 100%);
123     background: linear-gradient(right, rgba(0,0,0,0.05) 0%,rgba(0,0,0,0) 100%);
124 }
125 .view .s2 > .overlay {
126     background: -moz-linear-gradient(left, rgba(255,255,255,0) 0%, rgba(255, 255, 255, 0.2) 100%);
127     background: -webkit-linear-gradient(left, rgba(255,255,255,0) 0%, rgba(255, 255, 255, 0.2) 100%);
128     background: -o-linear-gradient(left, rgba(255,255,255,0) 0%, rgba(255, 255, 255, 0.2) 100%);
129     background: -ms-linear-gradient(left, rgba(255,255,255,0) 0%, rgba(255, 255, 255, 0.2) 100%);

```

```

130     background: linear-gradient(left, rgba(255,255,255,0) 0%, rgba(255, 255, 255, 0.2) 100%);
131 }
132 .view .s3 > .overlay {
133     background: -moz-linear-gradient(right, rgba(0,0,0,0.8) 0%, rgba(0,0,0,0.2) 100%);
134     background: -webkit-linear-gradient(right, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0.2) 100%);
135     background: -o-linear-gradient(right, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0.2) 100%);
136     background: -ms-linear-gradient(right, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0.2) 100%);
137     background: linear-gradient(right, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0.2) 100%);
138 }
139 .view .s4 > .overlay {
140     background: -moz-linear-gradient(left, rgba(0,0,0,0.8) 0%, rgba(0,0,0,0) 100%);
141     background: -webkit-linear-gradient(left, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0) 100%);
142     background: -o-linear-gradient(left, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0) 100%);
143     background: -ms-linear-gradient(left, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0) 100%);
144     background: linear-gradient(left, rgba(0,0,0,0.8) 0%,rgba(0,0,0,0) 100%);
145 }
146 .view .s5 > .overlay {
147     background: -moz-linear-gradient(left, rgba(0,0,0,0.3) 0%, rgba(0,0,0,0) 100%);
148     background: -webkit-linear-gradient(left, rgba(0,0,0,0.3) 0%,rgba(0,0,0,0) 100%);
149     background: -o-linear-gradient(left, rgba(0,0,0,0.3) 0%,rgba(0,0,0,0) 100%);
150     background: -ms-linear-gradient(left, rgba(0,0,0,0.3) 0%,rgba(0,0,0,0) 100%);
151     background: linear-gradient(left, rgba(0,0,0,0.3) 0%,rgba(0,0,0,0) 100%);
152 }
//HTML 代码
153 <div class="view">
154     <div class="slice s1" style="background-image: url(/.8.jpg);">
155         <span class="overlay"></span>
156         <div class="slice s2" style="background-image: url(/.8.jpg);">
157             <span class="overlay"></span>
158             <div class="slice s3" style="background-image: url(/.8.jpg);">
159                 <span class="overlay"></span>
160                 <div class="slice s4" style="background-image: url(/.8.jpg);">
161                     <span class="overlay"></span>
162                     <div class="slice s5" style="background-image: url(/.8.jpg);">
163                         <span class="overlay"></span>
164                     </div>
165                 </div>
166             </div>
167         </div>
168     </div>
169 </div>

```

本实例为了呈现 3D 立体效果，通过 s1 元素包含的 s2/s3/s4/s5 这 4 个后代元素来实现图片的层叠，以实现 3D 立体效果。在代码第 14~18 行，在包裹器 view 上设置 perspective，同时，在代码第 25~29 行设置 transform-style 为 preserve-3d，这两个属性是实现立体效果必

须同时设置的。在第 45~54 行设置后代元素 s2/s3/s4/s5 的 3D 效果。使用 `translate3d(x,y,z)` 使元素在 x/y/z 这 3 个维度中移动，也可以分开写这 3 个维度，如：`translateX(length)`、`translateY(length)`、`translateZ(length)`。注意 z 轴的值只能为 px。同理，使用 `scale3d(number,number,number)` 使得元素在这 x/y/z 3 个维度中缩放，也可分开写，如 `scaleX()`、`scaleY()`、`scaleZ()`。

代码第 95~116 行设置 `translate3d` 与 `rotate3d` 的不同值，使得图片呈现折叠效果。该 3D 图片呈现前后的具体效果如图 9.2 所示。



图 9.2 3D 图片

9.3 3D 按钮

为了实现 3D 按钮效果，我们将会利用 CSS 3 的 `box-shadow`、`linear-gradient`、`transition` 属性。本例 CSS 代码如下：

```
01 @font-face {                               /* 引入字体 */
02     font-family: 'EntypoRegular';
03     src: url('../font/entypo-webfont.eot');
04     src: url('../font/entypo-webfont.eot?#iefix') format('embedded-opentype'),
05          url('../font/entypo-webfont.woff') format('woff'),
06          url('../font/entypo-webfont.ttf') format('truetype'),
07          url('../font/entypo-webfont.svg#EntypoRegular') format('svg');
```

```

08     font-weight: normal;
09     font-style: normal;
10 }
11 .button, .button-bevel {
12     font-family: Arial, Helvetica, sans-serif;
13     font-size: 13px;
14     color: #fff;
15     text-decoration: none;
16     display: inline-block;
17     text-align: center;
18     padding: 7px 20px 9px;
19     margin: .5em .5em .5em 0;
20     cursor: pointer;
21     text-shadow: 0 1px 1px rgba(0,0,0,0.4);           /* 设置文字投影 */
22     -webkit-transition: 0.1s linear;                 /* 设置变换 */
23     -moz-transition: 0.1s linear;
24     -ms-transition: 0.1s linear;
25     -o-transition: 0.1s linear;
26     transition: 0.1s linear;
27 }
28 .button {
29     border-radius: 2px;
30     box-shadow: inset rgba(255,255,255,0.3) 1px 1px 0; /* 设置盒子投影 */
31 }
32 .button:active {
33     box-shadow: inset rgba(0,0,0,0.4) 0px 0px 6px;
34 }
35 /* 设置圆角按钮 */
36 .rounded {
37     border-radius: 20px;
38 }
39 .orange {                                           /* 设置背景颜色渐变，此处设置为橙色 */
40     background: rgb(255,183,0);
41     background: -moz-linear-gradient(top,  rgba(255,183,0,1) 0%, rgba(255,140,0,1) 100%);
42     background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgba(255,183,0,1)),
color-stop(100%,rgba(255,140,0,1)));
43     background: -webkit-linear-gradient(top,  rgba(255,183,0,1) 0%,rgba(255,140,0,1) 100%);
44     background: -o-linear-gradient(top,  rgba(255,183,0,1) 0%,rgba(255,140,0,1) 100%);
45     background: -ms-linear-gradient(top,  rgba(255,183,0,1) 0%,rgba(255,140,0,1) 100%);
46     background: linear-gradient(to bottom,  rgba(255,183,0,1) 0%,rgba(255,140,0,1) 100%);
47     filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ffb700', endColorstr=
'#ff8c00',GradientType=0 );
48     border: 1px solid #e59500;
49 }
50 .orange:hover {                                   /* 设置鼠标移动上的效果 */

```

```

51 background: rgb(255,203,72);
52 background: -moz-linear-gradient(top,  rgba(255,203,72,1) 0%, rgba(255,156,35,1) 100%);
53 background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgba(255,203,72,1)),
    color-stop(100%,rgba(255,156,35,1)));
54 background: -webkit-linear-gradient(top,  rgba(255,203,72,1) 0%,rgba(255,156,35,1) 100%);
55 background: -o-linear-gradient(top,  rgba(255,203,72,1) 0%,rgba(255,156,35,1) 100%);
56 background: -ms-linear-gradient(top,  rgba(255,203,72,1) 0%,rgba(255,156,35,1) 100%);
57 background: linear-gradient(to bottom,  rgba(255,203,72,1) 0%,rgba(255,156,35,1) 100%);
58 filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ffcb48', endColorstr=
    '#ff9c23',GradientType=0 );
59 }
60 .button-bevel {           /* 设置倾斜按钮的样式 */
61     vertical-align: top;
62     border-radius: 4px;
63     border: none;
64     padding: 10px 25px 12px;
65 }
66 .button-bevel:active {
67     position: relative;
68     top: 5px;
69 }
70 .button-bevel.orange {
71     box-shadow: #c46d00 0 6px 0px, rgba(0, 0, 0, 0.3) 0 10px 3px;
72 }
73 .button-bevel.orange:active {
74     box-shadow: #c46d00 0 3px 0, rgba(0, 0, 0, 0.2) 0 6px 3px;
75 }
76 .button-bevel.magenta:active {
77     box-shadow: #ca075c 0 3px 0, rgba(0, 0, 0, 0.2) 0 6px 3px;
78 }
79 /*设置倾斜按钮不同颜色的样式*/
80 .button-bevel.cyan {
81     box-shadow: #1994d3 0 6px 0px, rgba(0, 0, 0, 0.3) 0 10px 3px;
82 }
83 .button-bevel.cyan:active {
84     box-shadow: #1994d3 0 3px 0, rgba(0, 0, 0, 0.2) 0 6px 3px;
85 }
86 .button-bevel.red {
87     box-shadow: #88180e 0 6px 0px, rgba(0, 0, 0, 0.3) 0 10px 3px;
88 }
89 .button-bevel.red:active {
90     box-shadow: #88180e 0 3px 0, rgba(0, 0, 0, 0.2) 0 6px 3px;
91 }
92 .button-bevel.black {
93     box-shadow: #000 0 6px 0px, rgba(0, 0, 0, 0.3) 0 10px 3px, inset rgba(255, 255, 255, 0.3) 0 0 3px;

```

```

94 }
95 .button-bevel.black:active {
96   box-shadow: #000 0 3px 0, rgba(0, 0, 0, 0.2) 0 6px 3px, inset rgba(255, 255, 255, 0.3) 0 0 3px;
97 }
98 .button-bevel.green {
99   box-shadow: #439230 0 6px 0px, rgba(0, 0, 0, 0.3) 0 10px 3px;
100 }
101 .button-bevel.green:active {
102   box-shadow: #439230 0 3px 0, rgba(0, 0, 0, 0.2) 0 6px 3px;
103 }
104 .button span, .button-bevel span {
105   font-family: 'EntypoRegular';
106   font-size: 20px;
107   font-weight: normal;
108   vertical-align: middle;
109   line-height: 0;
110   margin-right: .1em;
111 }
112 /* 设置图案 */
113 .refresh:after { content: "h"; font-size: 34px; }
114 .shuffle:after { content: "f"; font-size: 34px; }
115 .preview:after { content: "M"; font-size: 34px; }
116 .tea:after { content: "u"; font-size: 34px; }
117 .wifi:after { content: "T"; font-size: 34px; }
118 .locator:after { content: "O"; font-size: 34px; }
119
120 .rss:after { content: "S"; font-size: 34px; }
121 .cloud:after { content: "y"; font-size: 34px; }
122 .download:after { content: "w"; font-size: 34px; }
123 .trash:after { content: "l"; font-size: 34px; }
124 .rack:after { content: "t"; font-size: 34px; }
125 .map:after { content: "1"; font-size: 34px; }
126
127 .setting:after { content: "@"; font-size: 34px; }
128 .identity:after { content: "."; font-size: 34px; }
129 .navigation:after { content: "2"; font-size: 34px; }
130 .gallery:after { content: "p"; font-size: 34px; }
131 .email:after { content: "%"; font-size: 34px; }
132 .users:after { content: "+"; font-size: 34px; }
133
134 .calendar:after { content: "P"; font-size: 34px; }
135 .link:after { content: ">"; font-size: 34px; }
136 .time:after { content: "N"; font-size: 34px; }
137 .folder:after { content: "s"; font-size: 34px; }
138 .tag:after { content: "C"; font-size: 34px; }

```

```

139 .share:after { content: "5"; font-size: 34px; }
140
141 .lock:after { content: "U"; font-size: 34px; }
142 .unlock:after { content: "V"; font-size: 34px; }
143 .mic:after { content: "O"; font-size: 34px; }
144 .love:after { content: "6"; font-size: 34px; }
145 .star:after { content: "7"; font-size: 34px; }
146 .like:after { content: "8"; font-size: 34px; }
147
148 .phone:after { content: "!"; font-size: 34px; }
149 .flag:after { content: "?"; font-size: 34px; }
150 .adduser:after { content: "-"; font-size: 34px; }
151 .attach:after { content: ""'; font-size: 34px; }
152 .comment:after { content: "."; font-size: 34px; }
153 .edit:after { content: "&"; font-size: 34px; }
154
155 .upload:after { content: "v"; font-size: 34px; }
156 .storage:after { content: "x"; font-size: 34px; }
157 .photo:after { content: "D"; font-size: 34px; }
158 .help:after { content: "K"; font-size: 34px; }
159 .glass:after { content: "R"; font-size: 34px; }
160 .print:after { content: "<"; font-size: 34px; }
161
162 .gadget:after { content: ""'; font-size: 34px; }

```

HTML 代码如下：

```

01 <div data-for="beveled">
02     <p>
03         <a href="javascript:void(0)" class="button-bevel orange"> <span class="refresh">
04             </span> Button </a>
05         <a href="javascript:void(0)" class="button-bevel orange"> <span class="shuffle">
06             </span> Button </a>
07         <a href="javascript:void(0)" class="button-bevel orange"> <span class="preview">
08             </span> Button </a>
09         <a href="javascript:void(0)" class="button-bevel orange"> <span class="tea">
10             </span> Button </a>
11         <a href="javascript:void(0)" class="button-bevel orange"> <span class="wifi">
12             </span> Button </a>
13         <a href="javascript:void(0)" class="button-bevel orange"> <span class="locator">
14             </span> Button </a>
15     </p>
16 </div>
17
18 <div data-for="rectangle">
19     <p>

```



```

15      <a href="javascript:void(0)" class="button orange"> <span class="refresh"></span>
    Button </a>
16      <a href="javascript:void(0)" class="button orange"> <span class="shuffle"></span>
    Button </a>
17      <a href="javascript:void(0)" class="button orange"> <span class="preview"></span>
    Button </a>
18      <a href="javascript:void(0)" class="button orange"> <span class="tea"></span>
    Button </a>
19      <a href="javascript:void(0)" class="button orange"> <span class="wifi"></span>
    Button </a>
20      <a href="javascript:void(0)" class="button orange"> <span class="locator"></span>
    Button </a>
21  </p>
22 </div>
23 <div data-for="rounded">
24   <p>
25       <a href="javascript:void(0)" class="button rounded orange"> <span class="edit">
    </span> Button </a>
26       <a href="javascript:void(0)" class="button rounded orange"> <span class="upload">
    </span> Button </a>
27       <a href="javascript:void(0)" class="button rounded orange"> <span class="storage">
    </span> Button </a>
28       <a href="javascript:void(0)" class="button rounded orange"> <span class="photo">
    </span> Button </a>
29       <a href="javascript:void(0)" class="button rounded orange"> <span class="help">
    </span> Button </a>
30       <a href="javascript:void(0)" class="button rounded orange"> <span class="locator">
    </span> Button </a>
31   </p>
32 </div>

```

为了便于按钮扩展形状、颜色、圆角等效果，此 3D 按钮在设计中分别将按钮倾斜、矩形、圆角等样式提取出来做封装，其中倾斜为 button-bevel、圆角为 rounded，并在目标 HTML 代码中，根据按钮需要添加不同样式。例如，倾斜按钮添加 class 为 button-bevel 样式，圆角添加 rounded 样式，矩形则不添加 button-bevel。

该 3D 按钮呈现的具体效果如图 9.3 所示。

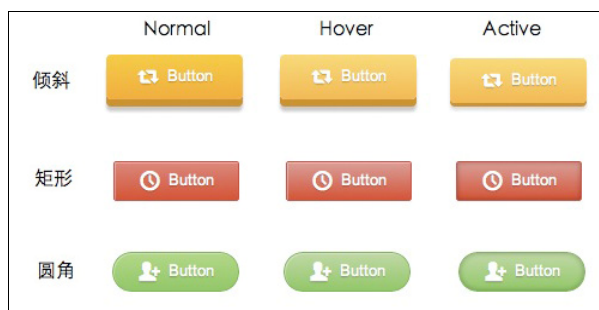


图 9.3 3D 按钮

9.4 3D 下拉菜单

与传统的下拉菜单不同，3D 下拉菜单的子菜单呈现垂直卷帘效果。为了实现此 3D 下拉菜单，需要使用 transition 的 translateZ 属性。实例代码如下：

```
//CSS 代码
01 a {
02     /* 贝叶斯曲线 */
03     -webkit-transition: all 250ms cubic-bezier(0.230, 1.000, 0.320, 1.000);
04     -moz-transition: all 250ms cubic-bezier(0.230, 1.000, 0.320, 1.000);
05     -ms-transition: all 250ms cubic-bezier(0.230, 1.000, 0.320, 1.000);
06     -o-transition: all 250ms cubic-bezier(0.230, 1.000, 0.320, 1.000);
07     transition: all 250ms cubic-bezier(0.230, 1.000, 0.320, 1.000);
08     text-decoration: none;
09 }
10 .header {
11     text-align: center;
12     position: absolute;
13     z-index: 1;
14     color: #333;
15     width: 100%;
16     top: 5%;
17 }
18 .header h1 {
19     letter-spacing: -1px;
20     text-shadow: -2px -1px 1px #fff, 1px 2px 2px rgba(0, 0, 0, 0.2);
21     font-weight: 300;
22     font-size: 36px;
23     margin: 0;
24 }
25 .header h2 {
26     text-transform: uppercase;      /* 设置文本为大写 */
27     text-shadow: -2px -1px 1px #fff, 1px 1px 1px rgba(0, 0, 0, 0.15);/* 设置文本投影 */
28     font-weight: 300;
29     font-size: 12px;
30     color: rgba(0,0,0,0.7);
31     margin: 0;
32 }
33 .demo:after {
34     box-shadow: 0 1px 16px rgba(0,0,0,0.15);
35     background: #1b1b1b;
36     position: absolute;
37     content: "";
38     height: 10px;
```

```
39     width: 100%;
40     top: 0;
41 }
42 /* 菜单列表样式 */
43 .list {
44     -webkit-transform-style: preserve-3d;
45     -moz-transform-style: preserve-3d;
46     -ms-transform-style: preserve-3d;
47     -o-transform-style: preserve-3d;
48     transform-style: preserve-3d;
49     text-transform: uppercase;
50     position: absolute;
51     margin-left: -140px;
52     top: 20%;
53 }
54 .list a {
55     display: block;
56     color: #fff;
57 }
58 .list a:hover {
59     text-indent: 20px;
60 }
61 .list dt, .list dd {
62
63     text-indent: 10px;
64     line-height: 55px;
65     background: #E0FBAC;
66     margin: 0;
67     height: 55px;
68     width: 270px;
69     color: #fff;
70 }
71 .list dt {
72     /* 3D 特效 */
73     -webkit-transform: translateZ(0.3px);
74     -moz-transform: translateZ(0.3px);
75     -ms-transform: translateZ(0.3px);
76     -o-transform: translateZ(0.3px);
77     transform: translateZ(0.3px);
78     text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.2);
79     font-size: 15px;
80 }
81
82 .list dd {
83     border-top: 1px dashed rgba(255,255,255,0.3);
```

```

84     line-height: 35px;
85     font-size: 11px;
86     height: 35px;
87     margin: 0;
88 }
89 /* 个性化颜色设置 */
90 .sashimi dt, .sashimi dd, .sashimi a { background: #73C8A9; }
91 .nigiri dt, .nigiri dd, .nigiri a { background: #E32551; }
92 .maki dt, .maki dd, .maki a { background: #FFC219; }
93 .sashimi a:hover { background: #61c19e; }
94 .nigiri a:hover { background: #d31b46; }
95 .maki a:hover { background: #ffbb00; }
96 .nigiri {
97     -webkit-transform: perspective(1200px) rotateY(40deg) !important;
98     -moz-transform: perspective(1200px) rotateY(40deg) !important;
99     -ms-transform: perspective(1200px) rotateY(40deg) !important;
100    -o-transform: perspective(1200px) rotateY(40deg) !important;
101    transform: perspective(1200px) rotateY(40deg) !important;
102
103    -webkit-transform-origin: 110% 25%;
104    -moz-transform-origin: 110% 25%;
105    -ms-transform-origin: 110% 25%;
106    -o-transform-origin: 110% 25%;
107    transform-origin: 110% 25%;
108
109    left: 20%;
110 }
111 .maki {
112     -webkit-transform: perspective(600px) translateZ(1px) !important;
113     -moz-transform: perspective(600px) translateZ(1px) !important;
114     -ms-transform: perspective(600px) translateZ(1px) !important;
115     -o-transform: perspective(600px) translateZ(1px) !important;
116     transform: perspective(600px) translateZ(1px) !important;
117
118     left: 50%;
119 }
120 .sashimi {
121     -webkit-transform: perspective(1200px) rotateY(-40deg) !important;
122     -moz-transform: perspective(1200px) rotateY(-40deg) !important;
123     -ms-transform: perspective(1200px) rotateY(-40deg) !important;
124     -o-transform: perspective(1200px) rotateY(-40deg) !important;
125     transform: perspective(1200px) rotateY(-40deg) !important;
126
127     -webkit-transform-origin: -10% 25%;
128     -moz-transform-origin: -10% 25%;

```

```

129     -ms-transform-origin: -10% 25%;
130     -o-transform-origin: -10% 25%;
131     transform-origin: -10% 25%;
132
133     left: 80%;
134 }
//HTML 代码
135 <section class="demo">
136     <dl class="list nigiri">
137         <dt>首页</dt>
138         <dd><a href="#">国内机票</a></dd>
139         <dd><a href="#">国际机票</a></dd>
140         <dd><a href="#">酒店</a></dd>
141         <dd><a href="#">客栈</a></dd>
142         <dd><a href="#">签证</a></dd>
143         <dd><a href="#">门票</a></dd>
144         <dd><a href="#">旅游度假</a></dd>
145         <dd><a href="#">手机版</a></dd>
146         <dd><a href="#">每日特惠</a></dd>
147         <dd><a href="#">发现航线</a></dd>
148     </dl>
149     <dl class="list maki">
150         <dt>机票</dt>
151         <dd><a href="#">我的机票</a></dd>
152         <dd><a href="#">查看机票</a></dd>
153         <dd><a href="#">机票预约</a></dd>
154         <dd><a href="#">票价历史</a></dd>
155         <dd><a href="#">航空保险</a></dd>
156         <dd><a href="#">明星商家</a></dd>
157         <dd><a href="#">往返航线</a></dd>
158         <dd><a href="#">单程航线</a></dd>
159         <dd><a href="#">国内低价</a></dd>
160         <dd><a href="#">航记攻略</a></dd>
161     </dl>
162     <dl class="list sashimi">
163         <dt>酒店</dt>
164         <dd><a href="#">国内酒店</a></dd>
165         <dd><a href="#">海外酒店</a></dd>
166         <dd><a href="#">港澳台</a></dd>
167         <dd><a href="#">品牌特价</a></dd>
168         <dd><a href="#">特价推荐</a></dd>
169         <dd><a href="#">国内酒店</a></dd>
170         <dd><a href="#">海外酒店</a></dd>
171         <dd><a href="#">游记攻略</a></dd>
172         <dd><a href="#">品牌商家</a></dd>

```

```
173         <dd><a href="#">消费者保障</a></dd>
174     </dl>
175 </section>
```

代码第 1~9 行，设置 transition 的 timing-function 属性值为 cubic-bezier，即设置了过渡效果的特效为贝叶斯曲线，cubic-bezier 即为贝叶斯曲线中的绘制方法。timing-function 预留的几个特效也可由 cubic-bezier 设置，具体如下：

```
01 //CSS 代码
02 ease: cubic-bezier(0.25, 0.1, 0.25, 1.0)
03 linear: cubic-bezier(0.0, 0.0, 1.0, 1.0)
04 ease-in: cubic-bezier(0.42, 0, 1.0, 1.0)
05 ease-out: cubic-bezier(0, 0, 0.58, 1.0)
06 ease-in-out: cubic-bezier(0.42, 0, 0.58, 1.0)
```

list 为菜单，dt 为菜单标题，dd 为菜单列表。设置 list 的 preserve-3d，子元素将保留其 3D 位置。在个性化设置的样式表中，即分别设置 class 为 nigiri、maki、sashimi 的样式中，设置 perspective 和 rotateY，实现卷帘效果。

该 3D 卷帘菜单的效果如图 9.4 所示，最终展示如图 9.5 所示。



图 9.4 3D 下拉菜单卷帘效果



图 9.5 3D 下拉菜单

9.5 3D 旋转动画

动画是使元素从一种样式逐渐变化为另一种样式的过程。在 CSS 3 中可以通过 `animation` 属性来实现动画。本节将介绍一种基于 HTML 5 的网页文字 3D 旋转动画效果，支持中文和英文字符，观看效果请注意要使用支持 CSS 3 技术的浏览器，技术主要是结合 `transition` 和 `animation` 来实现。

使用 `animation` 能够创建动画，这可以在许多网页制作中取代动画图片、Flash 动画以及 JavaScript。在 CSS 3 中创建动画，需要了解 `@keyframes` 规则，第 5 章已经介绍过动画的一些技术，这里只是简单回顾下。`@keyframes` 规则是用于创建动画时规定某项 CSS 属性的样式，就能创建由当前样式逐渐改为新样式的动画效果。在 `@keyframes` 中创建动画时，需将其绑定到某个选择器，否则不会产生动画效果。通过规定动画的名称及动画时长这两项 CSS 3 动画属性，即可将动画绑定到选择器。例如：

```
//CSS 代码
div {
    animation: myfirst 5s;
    -moz-animation: myfirst 5s;    /* Firefox */
    -webkit-animation: myfirst 5s; /* Safari 和 Chrome */
    -o-animation: myfirst 5s;     /* Opera */
}
```

动画的名称和时长是必选项，如果未设置动画时长，则动画的默认值是 0，不会进行动画效果。此外，还可以规定改变任意多的样式或任意多的次数，通过用百分比来规定变化发生的时间，或用关键词“from”和“to”，等同于 0%和 100%，0%是动画的开始，100%是动画的完成。为了得到最佳的浏览器支持，需始终定义 0%和 100%选择器。

实现 3D 旋转动画的实例代码如下：

```
//CSS 代码
01 .out_box {
02     width: 500px;
03     height: 300px;
04     margin: 100px auto 0;
05     overflow: hidden;
06 }
07 .out_box img {
08     float: left;
09 }
10 /* 设置 3D 动画参数 */
11 #animate_3d{
12     -webkit-perspective:600;
13     -webkit-transform-style:preserve-3d;
14     -webkit-animation-name:x-spin;
15     -webkit-animation-duration:7s;
16     -webkit-animation-iteration-count:infinite;
```

```

17     -webkit-animation-timing-function:linear;
18 }
19 .img_3d,.line_3d{
20     -webkit-transform-style:preserve-3d;
21     -webkit-animation-iteration-count:infinite;
22     -webkit-animation-timing-function:linear;
23 }
24 #animate_line_1{
25     -webkit-animation-name:y-spin;
26     -webkit-animation-duration:5s;
27 }
28 #animate_line_2{
29     -webkit-animation-name:back-y-spin;
30     -webkit-animation-duration:4s;
31 }
32 #animate_line_3{
33     -webkit-animation-name:y-spin;
34     -webkit-animation-duration:3s;
35 }
36 /* 设置@keyframes 参数规则 */
37 @-webkit-keyframes x-spin {
38     0%     { -webkit-transform:rotateX(0deg); }
39     50%    { -webkit-transform:rotateX(180deg); }
40     100%   { -webkit-transform:rotateX(360deg); }
41 }
42 @-webkit-keyframes y-spin {
43     0%     { -webkit-transform:rotateY(0deg); }
44     50%    { -webkit-transform:rotateY(180deg); }
45     100%   { -webkit-transform:rotateY(360deg); }
46 }
47
48 @-webkit-keyframes back-y-spin {
49     0%     { -webkit-transform:rotateY(360deg); }
50     50%    { -webkit-transform:rotateY(180deg); }
51     100%   { -webkit-transform:rotateY(0deg); }
52 }
//HTML 代码
53 <div id="animate_3d" class="out_box">
54     <div id="animate_line_1" class="line_3d">
55         
56         
57         
58         
59         
60     </div>

```



```

61     <div id="animate_line_2" class="line_3d">
62         
63         
64         
65         
66         
67     </div>
68     <div id="animate_line_3" class="line_3d">
69         
70         
71         
72         
73         
74     </div>
75 </div>

```

-webkit-perspective 表示透视范围大小，-webkit-transform-style 表示变换类型，preserve-3d 设置 3D 效果，-webkit-animation-name 表示动画名称，例如 x 轴旋转(x-spin)，y 轴旋转(y-spin)。-webkit-animation-duration 为动画持续的时间，单位为秒。-webkit-animation-iteration-count 表示动画循环的次数，默认为一次，参数 infinite 表示无穷次，即一旦开始实现动画效果，则一直执行，还可以将动画循环次数设置为任意的正整数，例如 animation-iteration-count:3，则动画循环 3 次。-webkit-animation-timing-function 即动画运动类型，参数有：ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier，这些参数归根结底是贝赛尔曲线（bezier）设置而来的。贝济埃曲线是应用于二维图形应用程序的数学曲线。曲线的定义有 4 个点：起始点、终止点（也称锚点）以及两个相互分离的中间点。滑动两个中间点，贝济埃曲线的形状会发生变化。

本例旋转之前的效果如图 9.6 所示，旋转时的效果如图 9.7 所示，旋转之后的效果如图 9.8 所示。



图 9.6 3D 旋转之前



图 9.7 3D 旋转时



图 9.8 3D 旋转之后

第 10 章 CSS Hack

由于不同的浏览器对 CSS 的解析各有差异，因此会导致不同的浏览器下生成的页面效果不一样，得不到实际所需要的页面效果。因此，在页面制作的过程中需要针对不同的浏览器设置不同 CSS 样式，使其能够同时兼容不同的浏览器。因此，解决不同浏览器对 CSS 解析的不同所采取的措施，一般称之为 CSS Hack，即对不同浏览器及其不同版本对 CSS 解析后出现页面内容的兼容性差异处理。

本章主要涉及的内容有：

- 让网站在所有浏览器下显示一致
- 解决 IE 6 中浮动元素的双倍边距问题
- 识别不同浏览器的方法
- 透明图片与透明背景的设置方法
- CSS 3 滤镜
- 常用的 CSS Hack 列表

10.1 让网站在所有浏览器下显示一致（CSS Reset）

由于浏览器的类型较多，且不同浏览器的默认样式也不同，在各种浏览器中，都会对 CSS 的选择器默认一些数值。譬如当 h1 没有被设置数值时，显示一定大小，但并不是所有的浏览器都使用一样的数值。因此，为了让网站在所有浏览器下显示一致，可以通过重新定义标签样式的方法，“覆盖”浏览器的 CSS 默认属性。为了便于复用，将这些用于“覆盖”浏览器默认样式的 CSS 内容统一抽离出来，便形成了 CSS Reset 文件，即用于重设浏览器样式的文件。在 reset.css 文件中将 HTML 元素的一些 style 定义一次，然后在需要的网页中引入这个 reset.css 文件即可。

目前比较流行的有 Eric Meyer 的重置样式和 YUI 的重置样式。另有 Condensed Meyer Reset 简化 Eric Meyer 的样式。在 YUI 的 reset 中，重置背景色为白色而文字颜色为黑色。

```
//CSS 代码
01  html {
02      color: #000;
03      background: #fff;
04  }
```

这里建议不要在 reset 中设置背景色，以防破坏用户的浏览器自定义网页默认背景色。

针对不同浏览器对具有内外边距元素的设置不同,有一种办法是通过定义* { margin: 0; padding: 0; }, 让所有元素的 padding 和 margin 都归零。在 YUI 的 reset 中,只把有 padding 和 margin 的元素样式清空,而其他元素不动,这样可以避免给一些无关元素带上不必要的样式,导致元素过多时的性能下降。

```
//CSS 代码
01  body, div, dl, dt, dd, ul, ol, li,
02  h1, h2, h3, h4, h5, h6, pre, code,
03  form, fieldset, legend, input, button,
04  textarea, p, blockquote, th, td {
05      margin: 0;
06      padding: 0;
07  }
```

同理,对于具有边框的元素做如下处理:

```
//CSS 代码
01  fieldset, img {
02      border: 0;
03  }
04  abbr, acronym {
05      border: 0;
06      font-variant: normal;
07  }
```

对于列表样式,YUI 的方式是:

```
//CSS 代码
01  li {
02      list-style: none;
03  }
```

Eric 的方式是:

```
//CSS 代码
01  ol, ul {
02      list-style: none;
03  }
```

这里建议使用设置 ol 和 ul 会比较稳妥。而且,波及的元素更少,性能会更高一点。虽然下载量会多 3 字节。

上标、下标以及 baseline 的重置:

```
//CSS 代码
01  sup, sub {
02      font-size: 100%;
03      vertical-align: baseline;
04  }
```

在某些浏览器中,q 或 blockquote 前后会出现引号,该引号严重破坏了美感,不受用户欢迎,所以需要重置:

```
//CSS 代码
```

```

01  blockquote, q {
02      quotes: none;
03  }
04  blockquote:before, blockquote:after,
05  q:before, q:after {
06      content: "";
07      content: none;
08  }

```

对于表格的样式有如下设置，需要在 html 中设置 `cellspacing=0` 达到 table 重置效果，且设置 `caption` 和 `th` 元素不要居中。

```

//CSS 代码
01  /* tables still need 'cellspacing="0"' in the markup */
02  table {
03      border-collapse: collapse;
04      border-spacing: 0;
05  }
06  caption, th {
07      text-align: left;
08  }

```

对于链接，在 YUI 中没有采取样式重置。这里建议把链接的下划线重置可以归纳进来，对整体项目有统一的链接样式。对个别元素的特殊设计可在后续的过程中个性开发。

```

//CSS 代码
01  a {
02      text-decoration: none;
03  }
04  :link, :visited {
05      text-decoration: none;
06  }

```

除了重置一些元素样式之外，这里目前主要推荐放入 `.clearfix`。清除浮动很重要，但这不属于样式重置，可放在布局中。

```

//CSS 代码
01  .clearfix:after {
02      content: ".";
03      display: block;
04      height: 0;
05      clear: both;
06      visibility: hidden;
07  }
08  .clearfix {display: inline-block;}
09  /* Hides from IE-mac */
10  * html .clearfix {height: 1%;}
11  .clearfix {display: block;}
12  /* End hide from IE-mac */

```

样式重置可以作为所有项目可以使用的共性存在，而不同的项目也可有其个性。当然还有其他一些共性，不属于样式重置的部分，但同样重要。由此，可引申为 CSS 框架。CSS 框架所要考虑的内容比一个 CSS Reset 要考虑的内容多，这里只是点到为止，不做更多展开，开发者可根据项目需要进行特性化定制。

10.2 解决 IE 6 中浮动元素的双倍边距问题

IE6 中，首个浮动到父元素边上的元素，如果具有该方向的 margin 值，margin 值会以双倍显示。如图 10.1 与图 10.2 所示，浮动元素的左边距在 IE 6 上为所设定的左边距的两倍。这个问题只会发生在浮动行的第一个浮动元素，准确的说法是每一行的第一个元素都会受此问题的影响。为了解决该问题，为浮动元素添加属性 `display: inline` 即可解决。

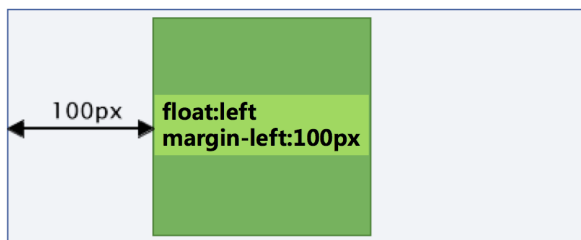


图 10.1 非 IE 6 单边距

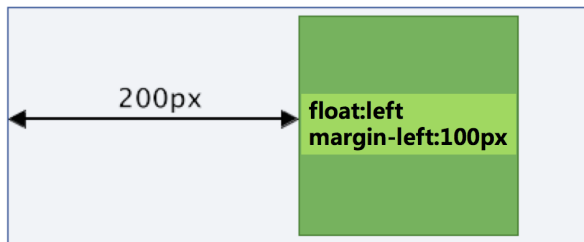


图 10.2 IE 6 双边距

下面讨论此 IE 6 浮动元素的双边距问题的延展性。

由图 10.3 和图 10.4 可以看出，当浮动元素 `#item2` 的 `margin-left` 小于等于 `#item1` 的宽度时显示正常，可是一旦大于 `#item1` 的宽度后，实际 `#item2` 的左边界距离等于 `#item1.width + (#item2.margin-left - #item1.width) * 2`。



图 10.3 margin-left 为 100px



图 10.4 margin-left 为 150px

要解决这个问题其实很简单,与双倍距离解决的方法一样,将#item2 设置 display: inline 即可。

10.3 识别不同浏览器

在开发页面时,常常会遇到所设计的网页中一些地方在各大浏览器(如 IE6、IE 7、IE 8、Firefox、Chrome 等)里有一些不同,如宽度、高度等有所差异。为了独立控制某种浏览器中的样式,常常会用 CSS 来识别不同浏览器,通过 CSS Hack,分别设计只能被特定浏览器识别的代码,可以达到这个目的。识别不同浏览器,不仅可以有效控制 CSS 代码在不同浏览器的表现,也可避免撰写多个 CSS 文件。

通常,在 IE 浏览器中,我们可以通过使用条件注释的 if 语句,来实现兼容不同版本的 IE 浏览器的写法。例如:

```

01 //CSS 代码
02 //仅所有的 Windows 系统自带 IE 可识别
03 <!--[if IE]>
04 Only IE CSS Style
05 <![endif]-->
06 //只有 IE 5 可以识别
07 <!--[if IE 5.0]>
08 Only IE 5.0
09 <![endif]-->
10 // IE 5 包换 IE 5.5 都可以识别
11 <!--[if gt IE 5.0]>
12 Only IE 5.0+
13 <![endif]-->
14 //仅 IE 6 可识别
15 <!--[if lt IE 6]>
16 Only IE 6-
17 <![endif]-->
18 // IE 6 以及 IE 6 以下的 IE 5.x 都可识别
19 <!--[if gte IE 6]>
20 Only IE 6/+

```

```

21 <![endif]-->
22 //仅 IE 7 可识别
23 <!--[if lte IE 7]>
24 Only IE 7/-
25 <![endif]-->
26 // IE 7 以及 IE 7 以下的 IE 6、IE 5.x 都可识别
27 <!--[if gte IE 7]>
28 Only IE 7/+
29 <![endif]-->
30 //仅 IE 8 可识别
31 <!--[if IE 8]>
32 Only IE 8/-
33 <![endif]-->

```

以上代码列举出了使用 CSS if Hack 方法针对 IE 浏览器及其不同版本所做的 Hack。

虽然 CSS if Hack 的方法简单有效，但是，有时我们为了局部代码需要做兼容性，设置 CSS if Hack 需要一大段烦琐的代码设置。为了简单地进行局部代码 CSS Hack，我们有其他的一些技巧来实现。

首先，通过特殊符号识别浏览器。例如，符号 *，只能被 IE 浏览器识别，其他浏览器（如 Firefox、Opera、Chrome 等）均不能识别 * 符号。如有以下代码：

```

01 //CSS 代码
02 #mydiv{
03     color:red;
04     *color:blue;           //使用特殊字符
05 }

```

那么元素#mydiv 在 Firefox 等非 IE 内核浏览器中，文字将呈现红色；而在 IE 中文字将呈现蓝色。

其次，通过!important 标识可以识别不同版本的 IE 浏览器。其实现原理是 IE 7 不但能识别 * 符号，还能识别!important，而 IE 6 只能识别 * 符号。如有以下代码：

```

01 //CSS 代码
02 #mydiv {
03     color: red !important; //使用特殊标识
04     *color: blue;         //使用特殊字符
05 }

```

以上代码在 IE 7 浏览器中，文字呈现红色；而在 IE 6 中呈现蓝色。

根据 * 符号与!important 的区别，可设计如下代码分别识别 Firefox、IE 6、IE 7：

```

01 //CSS 代码
02 #mydiv {
03     color: blue;
04     *color: red !important; //使用特殊标识
05     *color: green;         //使用特殊字符
06 }

```

以上代码在 Firefox 中，文字呈现蓝色，在 IE 7 浏览器中，呈现红色；而 IE 6 中呈现

蓝色。IE 核心的浏览器能识别*html 和*+html，而 Firefox 等非 IE 核心浏览器不能识别。

例如在 Firefox、IE 7、IE 6 中呈现 3 种不同文字颜色：

```
//CSS 代码
01  #div {
02      color: red;
03  }
04  *html #div {
05      color: green;
06  }
07  *+html #div {
08      color: blue;
09  }
```

此段代码中，第 1~3 行代码 Firefox 等浏览器可以正常识别，所以这些浏览器中文字呈红色；第 4~6 行 IE 6 能浏览器能正常识别并执行，用于针对 IE 6 独立写的样式，文字绿色；第 7~9 行只有 IE 7 才能正确识别，而 IE 6 和其他非 IE 核心浏览器不能，文字呈蓝色。

由于浏览器版本更新速度非常快，浏览器新版本的出现，使得需要设置兼容的代码更加复杂化，仅 IE 一种浏览器就需要兼容 IE 6~IE 10 共 5 个版本，尤其对 IE 8、9、10 这 3 个版本直接分辨还是很困难的。因此不仅仅需要在 Hack 上下工夫，还需使用选择器的部分技巧。

如以下代码便可以区分 IE 6~IE 10，以及 Opera、Chrome、Firefox 等浏览器。

```
01  //CSS 代码
02  #ie6 {                                /*for IE 6*/
03      _display:block;
04  }
05  #ie7 {                                /*for IE 7*/
06      *+display:block;
07  }
08  #ie8 {                                /*for IE 8*/
09      display:block\0;
10  }
11  :root #ie9 {                          /*for IE 9、10*/
12      display:block\0;
13  }
14  :root #ie8 {
15      display:none\0;
16  }
17  @media screen and (-ms-high-contrast: active), (-ms-high-contrast: none) {
18      /*for IE 10*/
19      :root #ie9,:root #ie8{
20          display:none;
21      }
22      :root #ie10 {
23          display:block;
```

```

24     }
25 }
26 @media screen and (-webkit-min-device-pixel-ratio:0) {
27 /* style for Chrome,safari...*/
28     #webkit {
29         display:block;
30     }
31 }
32 doesnotexist:-o-prefocus, #opera {
33 /* Opera 12+*/
34     display:block;
35 }
36 doesnotexist:-o-prefocus,.root #ie9 {
37 /* Opera 12+*/
38     display:none;
39 }
40 @-moz-document url-prefix() {
41 /*style for Firefox*/
42     #moz {
43         display:block;
44     }
45 }
//HTML 代码
46 <div id="ie6" class="global">
47     <p>Hi,你好! 我是 IE 6。</p>
48 </div>
49 <div id="ie7" class="global">
50     <p>Hi,你好! 我是 IE 7。</p>
51 </div>
52 <div id="ie8" class="global">
53     <p>Hi,你好! 我是 IE 8。</p>
54 </div>
55 <div id="ie9" class="global">
56     <p>Hi,你好! 我是 IE 9。</p>
57 </div>
58 <div id="ie10" class="global">
59     <p>Hi,你好! 我是 IE 10。</p>
60 </div>
61 <div id="opera" class="global">
62     <p>Hi,你好! 我是 Opera。</p>
63 </div>
64 <div id="webkit" class="global">
65     <p>Hi,你好! 我是 Webkit 内核的浏览器。</p>
66 </div>
67 <div id="moz" class="global">

```

```
68      <p>Hi,你好！我是 Firefox。</p>
69    </div>
```

代码第 9 行，使用\0 来识别 IE 8，由于\0 对所有 IE 8+都是支持的，故通过 IE 8 不能识别的:root 选择器来实现在高版本的 IE 浏览器中隐藏。:root 也被 IE 10 识别，故使用 IE 10 特用的 media 选择器来识别出 IE 10，并在其中也让 IE 9 隐藏，所以在第 20 行设置的 display: none，能达到分辨 IE 8、IE 9、IE 10 的目的。

又由于:root 选择器也被 Opera 浏览器识别，因此在为 Opera 浏览器指定样式时，要注意前面使用:root 的部分内容。即在第 34 行设置 display: block，在第 38 行设置 display: none，以区分出 Opera 浏览器。

10.4 背景与图片透明

本节介绍通过 CSS 实现层与背景半透明效果的设计方法。传统的实现方法为了达到透明的目的，通常背景使用半透明图片来实现。透明图片需使用 PNG 格式的图片。通过利用 PNG(Portable Network Graphics)图片的透明或半透明的特性能做出非常漂亮的网页来。在 Firefox、Chrome、Opera 等浏览器中对 PNG 的支持非常好，但是在 IE 6 浏览器上却无法显示这一 PNG 图片。虽然 IE7 已经支持，但 IE6 还是不支持。

为了让 IE 6 实现 PNG 图片透明，也可以通过 CSS 代码的方法实现。语法如下，参数见表 10.1。

```
01 //CSS 代码
02 filter : progid: DXImageTransform.Microsoft.AlphaImageLoader ( enabled=bEnabled ,
    sizingMethod=sSize , src=sURL )
```

表 10.1 filter属性参数

属性值	属性值的作用
enabled	可选项，布尔类型（Boolean），用于设置或检索滤镜是否激活，取值：true false true: 默认值。滤镜激活 false: 滤镜被禁止
sizingMethod	可选项，字符串类型（String），用于设置或检索滤镜作用的对象图片在对象容器边界内的显示方式
crop	用于剪切图片以适应对象尺寸
image	默认值。增大或减小对象的尺寸边界以适应图片的尺寸
scale	缩放图片以适应对象的尺寸边界
src	必选项，字符串类型(String)。使用绝对或相对URL地址指定背景图像。假如忽略此参数，滤镜将不会有作用

使用 filter 属性，可在对象容器边界内，在对象的背景和内容之间显示一张图片，并提供对此图片的剪切和改变尺寸的操作，如果载入的是 PNG 格式，则可设置 0%~100%的透明度。PNG 格式的图片的透明度不妨碍选择文本，即设置的时候可以选择显示在 PNG 格式的图片完全透明区域后面的内容。

设置 IE 6 实现 PNG 图片透明的实例代码：

```

01 //CSS 代码
02 #div1 {
03     height: 600px;
04     width: 260px;
05     padding: 20px;
06     background-repeat: repeat;
07 }
08 html>body #div1 {
09     background-repeat: repeat;
10     background-image: url(bj1.png);
11 }
12 * #div1 {
13     filter: progid:DXImageTransform.Microsoft.AlphaImageLoader(enabled=true, sizingMethod=
scale, src="bg.png")
14 }
```

Firefox、Opera 等完全支持 PNG 透明图片的浏览器也支持子选择器 (>)，而 IE 不识别，所有可以通过定义 CSS Hack 的方式，针对性地设置 Firefox、Opera 等浏览器中 PNG 图片的样式。需要注意的是 AlphaImageLoader 滤镜会导致该区域的链接和按钮无效，解决的办法是为链接或按钮添加代码 position: relative，使其相对定位。AlphaImageLoader 无法设置背景的重复，所以对图片的切图精度会有很高的精确度要求。

另一种 IE6 中 PNG 透明背景图片处理方法的实例代码如下：

```

01 //CSS 代码
02 .png{
03     background: url(/angel.png) no-repeat !important;
04     _filter: progid:DXImageTransform.Microsoft.AlphaImageLoader(src="/images/angel.png");
05     background:none;
06     width:100px;
07     height:100px;
08 }
09 //HTML 代码
10 <div class="png">背景 PNG 透明</div>
```

事实上，通过 CSS，也能达到半透明的目的。在 IE 浏览器中，可以通过 filter 属性来设置透明：

```

01 //CSS 代码
02 div {
03     filter: alpha(Opacity=80);
04     -moz-opacity: 0.5;
05     -webkit-opacity: 0.5;
06     -o-opacity: 0.5;
07     opacity: 0.5;
08 }
09 //HTML 代码
```

```

10 <div class="div-a">
11     <div class="div-b">实现 DIV 半透明实例演示</div>
12 </div>

```

以上代码中，第3行通过使用 filter 属性对 Windows 下的 IE 浏览器设置半透明滤镜效果，filter: alpha(Opacity=80)表示设置该对象 80%半透明，Firefox、Chrome、Opera 等浏览器无法识别 filter 属性，但是可以通过 opacity 属性来设置半透明。代码第4~7行是通过设置对应浏览器的 opacity 属性，opacity: 0.5 相当于设置 50%半透明。未设置透明和设置透明的效果分别如图 10.5 和图 10.6 所示。

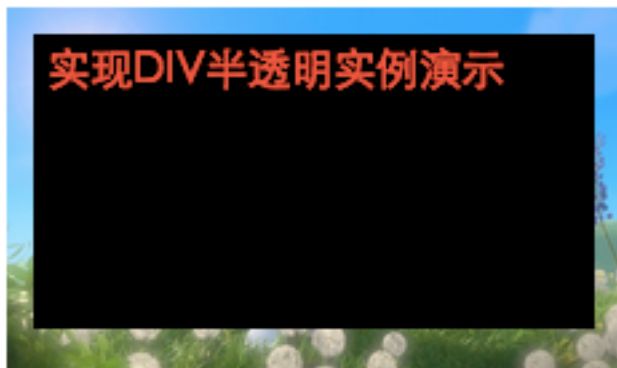


图 10.5 未设置透明



图 10.6 已设置透明的效果

通过对比图 10.5 与图 10.6 可以看出，图 10.6 通过设置 opacity 属性达到了设置元素背景透明的效果。但是，该实现方案的缺点是，该背景透明的元素内部的文字也相应地呈现出透明度。为了使得元素背景透明的同时，其内部的文字呈现非透明，需要单独将文字提取出来单独放置在与透明元素同级的位置，通过绝对定位来达到目的。代码如下：

```

01 //CSS 代码
02 .div-a {
03     background: url(../images/div-a-bg.jpg) no-repeat;
04     background-size: 230px 136px;
05     width: 230px;
06     height: 136px;

```

```

07     padding: 10px;
08     position: relative;
09 }
10 .div-b {
11     background: #000;
12     width: 200px;
13     height: 100px;
14     filter: alpha(Opacity=60);
15     -moz-opacity: 0.6;
16     opacity: 0.6;
17     position: absolute;
18     top: 10px;
19     left: 10px;
20 }
21 .div-b-content {
22     width: 200px;
23     height: 100px;
24     color: #F00;
25     position: absolute;
26     top: 10px;
27     left: 10px;
28     text-align: center;
29     padding-top: 5px;
30 }
//HTML 代码
31 <div class="div-a">
32     <div class="div-b"></div>
33     <div class="div-b-content">实现 DIV 半透明实例演示</div>
34 </div>

```

实现效果如图 10.7 所示。



图 10.7 设置背景透明、内容不透明

10.5 IE 10 CSS Hack

由于 Windows 8 中绑定的是 IE 10 浏览器，Windows 8 上市后，需要检测网页对 IE 10 的兼容性。但是往往发现之前的许多页面在 IE 10 中表现不正常，其原因有些是对 IE 不同版本的 Hack 引发的，有些不确定是否是 IE 10 引出的呈现问题，所以需要重新针对 IE 10 进行 CSS Hack。

首先，IE 10 不支持 CSS 条件注释。因此，针对 IE 10，可以额外利用 IE 私有的条件编译（conditional compilation）结合条件注释来提供 CSS Hack：

```
//CSS 代码
01 <!--[if !IE]><!--<script>
02 if (/*@cc_on!@*/false) {
03     document.documentElement.className+=' ie10'; //样式名：IE 10
04 }
05 </script><!--<![endif]-->
```

通过特效检测/*@cc_on!@*/即可在 IE 10 中给 html 元素添加一个 class=“ie10”，然后针对 IE 10 的样式可以写在该选择器内部：

```
01 /*CSS 代码*/
02 .ie10 .example {
03     /* 仅写支持 IE 10 的样式表 */
04 }
```

虽然条件编译支持 IE 浏览器的所有版本，但是其他浏览器暂不支持。需要注意的是，条件编译不支持在 Windows Store 中的 APP 里使用，只支持在 IE 浏览器中使用。此外，也可以通过 JavaScript 在浏览器头部 User Agent 检测出 IE 10 版本，并在 IE 10 浏览器的 html 元素添加 class 的方法来实现。

其次，IE10 支持媒体查询，并且也同样支持-ms-high-contrast 属性，因此可以利用这个特性来针对 IE 10 来进行 Hack。

```
//CSS 代码
01 @media screen and (-ms-high-contrast: active), (-ms-high-contrast: none) {
02     /* IE10-specific styles go here */
03 }
```

这种写法可以适配到高对比度和默认模式，因此可以覆盖到所有 IE 10 的模式了，并且这种方式可能也会在后续的 IE 11 中生效。如果通过 JavaScript 的方法，可以同时结合条件编译与媒体查询来进行 IE 10 的 Hack：

```
01 if (window.matchMedia("screen and (-ms-high-contrast: active), (-ms-high-contrast: none)").
matches) {
02     document.documentElement.className += "ie10";
03 }
```

10.6 CSS 3 滤镜

CSS 3 Filters 是区别于 CSS filter 的 CSS 3 滤镜，更不是 IE 滤镜。CSS 3 Filters 主要是运用在图片上，以实现一些特效。尽管 CSS 3 Filters 也能运用在视频的处理上，但是本节主要讨论在图片上的运用。

CSS 3 Filters 语法：

```
//CSS 代码
01  elm {
02      filter: none | <filter-function> [ <filter-function> ]*
03  }
```

filter 的默认值是 none，不具备继承性，即子元素不会继承父元素的滤镜方案设置。filter 的参数值中 filter-function 参数见表 10.2。

表 10.2 CSS 3 filter-function 参数可选值

属性值	属性值的作用
grayscale	灰度，默认值：100%
sepia	老照片，默认值：100%
saturate	饱和度，默认值：100%
hue-rotate	色相旋转，默认值：0 deg
invert	反色，默认值：100%
opacity	透明度，默认值：100%
brightness	亮度，默认值：100%
contrast	对比度，默认值：100%
blur	模糊，默认值：0
drop-shadow	阴影

从表 10.2 中可以看出，filter-function 的参数值与 Photoshop 中对图片的处理参数几乎类似。

实例代码如下：

```
//CSS 代码
01  /*设置灰度滤镜*/
02  .greyscale img {
03      filter: grayscale(1);
04      -webkit-filter: grayscale(1);
05      -moz-filter: grayscale(1);
06      -o-filter: grayscale(1);
07      -ms-filter: grayscale(1);
08  }
09  .greyscale img:hover {
10      filter: grayscale(0);
11      -webkit-filter: grayscale(0);
12      -moz-filter: grayscale(0);
13      -o-filter: grayscale(0);
```



```
14     -ms-filter: grayscale(0);
15 }
16 /*设置老照片滤镜*/
17 .sepia img {
18     filter: sepia(1);
19     -webkit-filter: sepia(1);
20     -moz-filter: sepia(1);
21     -o-filter: sepia(1);
22     -ms-filter: sepia(1);
23 }
24 .sepia img:hover {
25     filter: sepia(0);
26     -webkit-filter: sepia(0);
27     -moz-filter: sepia(0);
28     -o-filter: sepia(0);
29     -ms-filter: sepia(0);
30 }
31 /*设置饱和度滤镜*/
32 .saturate img {
33     filter: saturate(500%);
34     -webkit-filter: saturate(500%);
35     -moz-filter: saturate(500%);
36     -o-filter: saturate(500%);
37     -ms-filter: saturate(500%);
38 }
39 .saturate img:hover {
40     filter: saturate(100%);
41     -webkit-filter: saturate(100%);
42     -moz-filter: saturate(100%);
43     -o-filter: saturate(100%);
44     -ms-filter: saturate(100%);
45 }
46 /*设置色相反转滤镜*/
47 .hue-rotate img {
48     filter: hue-rotate(180deg);
49     -webkit-filter: hue-rotate(180deg);
50     -moz-filter: hue-rotate(180deg);
51     -o-filter: hue-rotate(180deg);
52     -ms-filter: hue-rotate(180deg);
53 }
54 .hue-rotate img:hover {
55     filter: hue-rotate(0);
56     -webkit-filter: hue-rotate(0);
```

```

57     -moz-filter: hue-rotate(0);
58     -o-filter: hue-rotate(0);
59     -ms-filter: hue-rotate(0);
60 }
61 /*设置反色滤镜*/
62 .invert img {
63     filter: invert(1);
64     -webkit-filter: invert(1);
65     -moz-filter: invert(1);
66     -o-filter: invert(1);
67     -ms-filter: invert(1);
68 }
69 .invert img:hover {
70     filter: invert(0);
71     -webkit-filter: invert(0);
72     -moz-filter: invert(0);
73     -o-filter: invert(0);
74     -ms-filter: invert(0);
75 }
76 /*设置透明度滤镜*/
77 .opacity img {
78     opacity:0.3;
79 }
80 .opacity img:hover {
81     opacity:1;
82 }
83 /*设置亮度滤镜*/
84 .brightness img {
85     filter: brightness(50%);
86     -webkit-filter: brightness(50%);
87     -moz-filter: brightness(50%);
88     -o-filter: brightness(50%);
89     -ms-filter: brightness(50%);
90 }
91 .brightness img:hover {
92     filter: brightness(100%);
93     -webkit-filter: brightness(100%);
94     -moz-filter: brightness(100%);
95     -o-filter: brightness(100%);
96     -ms-filter: brightness(100%);
97 }
98 /*设置对比度滤镜*/
99 .contrast img {

```

```

100     filter: contrast(0.3);
101     -webkit-filter: contrast(0.3);
102     -moz-filter: contrast(0.3);
103     -o-filter: contrast(0.3);
104     -ms-filter: contrast(0.3);
105 }
106 .contrast img:hover {
107     filter: contrast(1);
108     -webkit-filter: contrast(1);
109     -moz-filter: contrast(1);
110     -o-filter: contrast(1);
111     -ms-filter: contrast(1);
112 }
113 /*设置模糊滤镜*/
114 .blur img {
115     filter: url(/static/svg/filter.svg#blur);
116     filter: blur(5px);
117     -webkit-filter: blur(5px);
118     -moz-filter: blur(5px);
119     -o-filter: blur(5px);
120     -ms-filter: blur(5px);
121 }
122 .blur img:hover {
123     filter: blur(0);
124     -webkit-filter: blur(0);
125     -moz-filter: blur(0);
126     -o-filter: blur(0);
127     -ms-filter: blur(0);
128 }
129 /*设置阴影*/
130 .drop-shadow{
131     -webkit-filter: drop-shadow(5px 5px 5px #ccc);
132 }
//HTML 代码
133 <div class="greyscale">
134     
135 </div>

```

其中，设置 Greyscale 属性会使图片变灰。属性值（value）既可以是分数，也可以是百分数，它的作用是使图片更灰或者不灰。代码第 2~15 行，对图片使用了 Greyscale 属性，使得图片灰度变化。Greyscale 滤镜设置前后的展示效果见图 10.8 与图 10.9 所示。

Sepia 滤镜对图片的处理效果，图片的显示类似于回到了五、六十年代的那种老照片。代码第 16~30 行，对图片使用了 Sepia 属性。

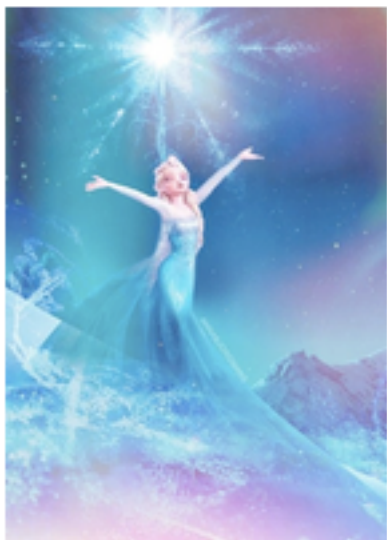


图 10.8 设置 Greyscale 滤镜之前



图 10.9 设置 Greyscale 滤镜之后

Saturate 效果滤镜可以增加图片的颜色饱和度。这里的属性值是饱和度的比例，例如百分之两百，若 **Saturate** 设置为百分之百则是不变。代码第 31~45 行，对图片使用了 **Saturate** 属性，使得图片饱和度发生变化。**Saturate** 滤镜设置前后的展示效果见图 10.10 与图 10.11 所示。



图 10.10 设置 Saturate 滤镜之前

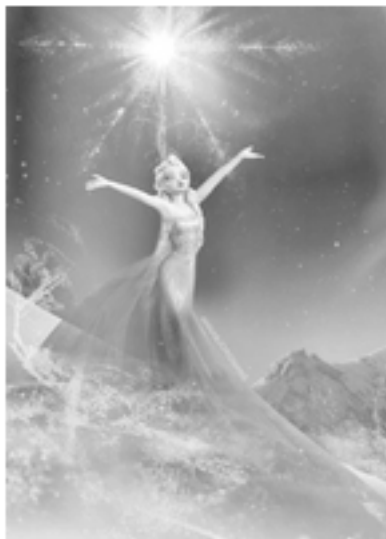


图 10.11 设置 Saturate 滤镜之后

Hue Rotate 滤镜是对色相进行旋转，通过色相的旋转，可以对图片的色彩进行重组。**Hue Rotate** 滤镜属性的取值为 360 度内的任意值。代码第 47~60 行，对图片使用了 **Hue Rotate** 属性，使得图片色相发生变化。

Invert 滤镜是反相效果，设置 **Invert** 滤镜使得图片看起来类似于老相机的底片效果一样。代码第 61~75 行，对图片使用了 **Invert** 属性，应用反相，使得图片的颜色色相反转。

Opacity 这个滤镜估计就比较熟悉了，是使图片透明。代码第 76~83 行，对图片使用了

Opacity 属性，修改图片的透明度。

Brightness 滤镜是对图片的亮度进行处理，属性值为百分比。代码第 84~97 行，对图片使用了 Brightness 属性，使得图片亮度发生变化。Contrast 滤镜是对图片最亮的地方和最暗的地方进行调整，从而改变图片的均衡感。代码第 98~112 行，对图片使用了 Contrast 属性，使得图片对比度发生变化，即图片的灰度反差的大小发生变化。

Blur 滤镜是使图片模糊。模糊的程度来自于设置的属性值。代码第 113~128 行，对图片使用了 Blur 属性，进行模糊。Blur 滤镜设置前后的展示效果见图 10.12 与图 10.13 所示。Drop-shadow 与 box-shadow 一样的效果，给图片加阴影效果。代码第 129~132 行，对图片使用了 Drop-shadow 属性，使得图片具有阴影。



图 10.12 设置 Blur 滤镜之前



图 10.13 设置 Blur 滤镜之后

注意：目前支持 filter-function 属性的浏览器较少，只是 Webkit 支持，而且只有 Webkit nightly 版本和 Chrome 18.0.976 以上版本支持。

10.7 常用的 CSS Hack 列表

在网页制作的过程中，由于不同浏览器的内核不同，有时需要对不同的浏览器分别设置 CSS Hack，以下总结了常用的 CSS Hack 便于查询，具体代码如下：

```
//CSS 代码
01  /* IE 6 及其以下版本 */
02  * html #uno {
03      color: red
04  }
05  /* IE 7 */
06  *:first-child+html #dos {
```

```

07     color: red
08 }
09 /* IE 7、 FF、 Saf、 Opera */
10 html>body #tres {
11     color: red
12 }
13 /* IE 8、 FF、 Safari、 Opera (所有浏览器除了 IE 6、 7) */
14 html>/**/body #cuatro {
15     color: red
16 }
17 /* Opera 9.27 及其以下版本、 Safari 2 */
18 html:first-child #cinco {
19     color: red
20 }
21 /* Safari 2-3 */
22 html[xmlns=""] body:last-child #seis {
23     color: red
24 }
25 /* Safari 3+、 Chrome 1+、 Opera 9+、 ff 3.5+ */
26 body:nth-of-type(1) #siete {
27     color: red
28 }
29 /* Safari 3+、 Chrome 1+、 Opera 9+、 ff 3.5+ */
30 body:first-of-type #ocho {
31     color: red
32 }
33 /* Safari 3+、 Chrome 1+ */
34 @media screen and (-webkit-min-device-pixel-ratio:0) {
35     #diez {
36         color: red
37     }
38 }
39 /* iPhone / mobile Webkit */
40 @media screen and (max-device-width: 480px) {
41     #veintiseis {
42         color: red
43     }
44 }
45 /* Safari 2 - 3.1 */
46 html[xmlns=""]:root #trece {
47     color: red
48 }
49 /* Safari 2 - 3.1、 Opera 9.25 */
50 *|html[xmlns=""] #catorce {
51     color: red
52 }

```

```

53 /* 所有浏览器，除了 IE6-8 */
54 :root *> #quince {
55     color: red
56 }
57 /* IE 7 */
58 *+html #dieciocho {
59     color: red
60 }
61 /* Firefox only. 1+ */
62 #veinticuatro, x:-moz-any-link {
63     color: red
64 }
65 /* Firefox 3.0+ */
66 #veinticinco, x:-moz-any-link, x:default {
67     color: red
68 }
69 /* IE 6 */
70 #once {
71     _color: blue
72 }
73 /* IE 6、IE 7 */
74 #doce {
75     *color: blue;
76 }
77 /* IE 6、IE 7 */
78 #doce {
79     #color: blue;
80 }
81 /* 所有浏览器除了 IE6 */
82 #diecisiete {
83     color/**/: blue
84 }
85 /* IE 6、IE 7、IE 8 */
86 #diecinueve {
87     color: blue\9;
88 }
89 /* IE 7、IE 8 */
90 #veinte {
91     color/*\*/: blue\9;
92 }
93 /* IE 6、IE 7 -- 与 !important 相同*/
94 #veintesiete {
95     color: blue !ie;
96 }

```

其中，第 73~80 行代码，分别使用符号*和符号#对 IE 6 与 IE 7 进行 Hack。

10.8 CSS 重置方案 (CSS Reset)

10.1 节介绍了几种简单的 Reset 方案,并介绍了针对边框、列表等具体元素所作的 Reset 代码,这里我们介绍 3 种全局 Reset 方案。

10.8.1 方案一

```
01  html, body, div, span, applet, object, iframe, table, caption,
02  tbody, tfoot, thead, tr, th, td, del, dfn, em, font, img, ins,
03  kbd, q, s, samp, small, strike, strong, sub, sup, tt, var,
04  h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr,
05  acronym, address, big, cite, code, dl, dt, dd, ol, ul, li,
06  fieldset, form, label, legend {
07      vertical-align: baseline;
08      font-family: inherit;
09      font-weight: inherit;
10      font-style: inherit;
11      font-size: 100%;
12      outline: 0;
13      padding: 0;
14      margin: 0;
15      border: 0;
16  }
17  :focus {
18      outline: 0;
19  }
20  body {
21      background: white;
22      line-height: 1;
23      color: black;
24  }
25  ol, ul {
26      list-style: none;
27  }
28  table {
29      border-collapse: separate;
30      border-spacing: 0;
31  }
32  caption, th, td {
33      font-weight: normal;
34      text-align: left;
35  }
36  blockquote:before, blockquote:after, q:before, q:after {
```



```

37     content: "";
38 }
39 blockquote, q {
40     quotes: "" "";
41 }

```

10.8.2 方案二（雅虎方案）

```

01 body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,
02 form,fieldset,input,textarea,p,blockquote,th,td{
03     padding: 0;
04     margin: 0;
05 }
06 table {
07     border-collapse: collapse;
08     border-spacing: 0;
09 }
10 fieldset,img {
11     border: 0;
12 }
13 address,caption,cite,code,dfn,em,strong,th,var{
14     font-weight: normal;
15     font-style: normal;
16 }
17 ol,ul {
18     list-style: none;
19 }
20 caption,th {
21     text-align: left;
22 }
23 h1,h2,h3,h4,h5,h6 {
24     font-weight: normal;
25     font-size: 100%;
26 }
27 q:before,q:after {
28     content:"";
29 }
30 abbr,acronym {
31     border:0;
32 }

```

10.8.3 方案三

这种方案因其全面性，尤其是支持最新的 HTML 5 中的某些标签重置而受人欢迎，如果前面的两种方案满足不了快速发展的 HTML 技术的需求，可以优先考虑这种方案，代码

上也很轻量。

```

01  html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr,
02  acronym, address, big, cite, code
03  , del, dfn, em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl,
04  dt, dd, ol, ul, li, fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article,
05  aside, canvas, details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output,
06  ruby, section, summary, time, mark, audio, video
07      { margin:0; padding:0; border:0; font-size:100%; font:inherit; vertical-align:baseline;
08      outline:none; }
09  html
10      { height:101%; } /* 总是显示滚动条 */
11  body
12      { font-size:62.5%; line-height:1; font-family:Arial, Tahoma, Verdana, sans-serif; }
13  article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section
14      { display:block; }
15  img
16      { border:0; max-width:100%; }
17  a
18      { text-decoration:none; }
19  a:hover
20      { text-decoration:underline; }
21  ol, ul
22      { list-style:none; }
23  blockquote, q
24      { quotes:none; }
25  blockquote:before, blockquote:after, q:before, q:after
26      { content:""; content:none; }
27  strong
28      { font-weight:bold; }
29  input
30      { outline:none; }
31  table
32      { border-collapse:collapse; border-spacing:0; }

```

使用 CSS 重置抹去所有浏览器默认属性，网页变为完全空白的画布，写 CSS 代码是向空白添加样式而不是修改样式，这样写代码更符合逻辑性和调理性。CSS 重置可以使不同浏览器的兼容问题降至最小的水平，从而提高开发效率。

注意：任何东西都有两面性，CSS 重置的添加使代码的体积变大，对于许多新手来说，使用过 CSS 重置之后可能带来修改样式这样更麻烦的工作，另外，浏览器的某些默认设置也并不是不可取的。在出现错误时，从空白构建起的 CSS 代码比使用过 CSS 重置的代码更容易定位错误。对于这个问题，笔者的建议是参考 CSS Reset，在实践中构建一套适合自己的 CSS 重置代码，最后在大项目中使用通用 CSS 代码时才会得心应手。

考虑到敲代码费时费力，所有的方案已经写入实例文件中，可以直接复制粘贴使用。

第 11 章 其他常用代码



在设计和实现页面的过程中，常常会遇到相似的显示效果或是一些可复用的特效，本节介绍一些常用的代码，便于搭建网站进行复用。

本章主要涉及的知识点有：

- 使用 CSS 3 实现简单的计算器、播放器、梯度模板
- 不使用 table 的 form 表单
- 可以重复利用的规则、在同一元素上使用多种类
- CSS 块引用模版
- 一般媒体查询与响应式设计
- 花式 CSS 3 Pull-引文
- 关于字符编码问题
- 一些手机 APP 的简洁设计
- 禁用 Webkit 内核某些属性

11.1 使用 CSS 3 实现简单的计算器

本节介绍如何使用 CSS 3 技术实现简单的彩色计算器效果。计算器由包括显示区与键盘组成。键盘的按键包括功能键与数字键。其中，功能键包括计算符与操作符，计算符即加、减、乘、除，分别用+、-、*、/ 显示；操作符包括清除键、等于符。该 CSS 3 简单计算器的最终效果如图 11.1 所示。

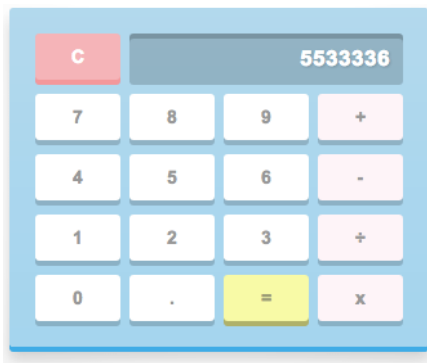


图 11.1 CSS 3 简单计算器

实例 HTML 代码如下：

```

01 <div id="calculator">
02     <!-- Screen and clear key -->
03     <div class="top">
04         <span class="clear">C</span>
05         <div class="screen"></div>
06     </div>
07     <div class="keys">
08         <!-- operators and other keys -->
09         <span>7</span>
10         <span>8</span>
11         <span>9</span>
12         <span class="operator">+</span>
13         <span>4</span>
14         <span>5</span>
15         <span>6</span>
16         <span class="operator">-</span>
17         <span>1</span>
18         <span>2</span>
19         <span>3</span>
20         <span class="operator">÷</span>
21         <span>0</span>
22         <span>.</span>
23         <span class="eval">=</span>
24         <span class="operator">x</span>
25     </div>
26 </div>

```

根据计算器的特点，在 HTML 代码结构设计的时候，也遵从计算器的结构，即分为显示区域与操作区域。在 HTML 代码的第 3~6 行是显示区，第 7~25 行是操作区。

CSS 代码如下：

```

01 /* 基本浏览器重置 */
02 * {
03     margin: 0;
04     padding: 0;
05     box-sizing: border-box;
06     font: bold 14px Arial, sans-serif; /* 设置字体 */
07 }
08 /*
09 * 背景渐变
10 * radial-gradient 径向渐变
11 */
12 html {
13     height: 100%;
14     background: white;

```

```

15     background: radial-gradient(circle, #fff 20%, #ccc);
16     background-size: cover;
17 }
18 /* 使用 box-shadow, 创造 3D 特效 */
19 #calculator {
20     width: 325px;
21     height: auto;
22     margin: 100px auto;
23     padding: 20px 20px 9px;
24     background: #9dd2ea;
25     /* linear-gradient 线性渐变 */
26     background: linear-gradient(#9dd2ea, #8bceec);
27     border-radius: 3px;
28     /* 盒子阴影 */
29     box-shadow: 0px 4px #009de4, 0px 10px 15px rgba(0, 0, 0, 0.2);
30 }
31 /* 顶部 */
32 .top span.clear {
33     float: left;
34 }
35 /* 使用内阴影 inset, 创造缩进效果 */
36 .top .screen {
37     height: 40px;
38     width: 212px;
39     float: right;
40     padding: 0 10px;
41     background: rgba(0, 0, 0, 0.2);
42     border-radius: 3px;
43     /* 设置盒阴影 */
44     box-shadow: inset 0px 4px rgba(0, 0, 0, 0.2);
45     /* 设置字体、排版 */
46     font-size: 17px;
47     line-height: 40px;
48     color: white;
49     text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.2);
50     text-align: right;
51     letter-spacing: 1px;
52 }
53 /* 清除浮动 */
54 .keys, .top {
55     overflow: hidden;
56 }
57 /*
58 * 设置键盘统一样式
59 * 键盘统一设置为向左浮动、相对定位、鼠标停留手型、文字居中等

```

```

60  */
61  .keys span, .top span.clear {
62      float: left;
63      position: relative;
64      top: 0;
65      cursor: pointer;
66      width: 66px;
67      height: 36px;
68      background: white;
69      border-radius: 3px;
70      box-shadow: 0px 4px rgba(0, 0, 0, 0.2);
71      margin: 0 7px 11px 0;
72      color: #888;
73  /* 设置统一的行高 */
74      line-height: 36px;
75      text-align: center;
76  /* 设置内容不可选 */
77      user-select: none;
78  /* 使用 CSS 3 transitions, 使得键盘样式变化更为平滑 */
79      transition: all 0.2s ease;
80  }
81  /* 设置操作符、运算符、清除等按键的特殊样式 */
82  .keys span.operator {
83      background: #FFF0F5;
84      margin-right: 0;
85  }
86  .keys span.eval {
87      background: #f1ff92;
88  /* 设置盒阴影 */
89      box-shadow: 0px 4px #9da853;
90      color: #888e5f;
91  }
92  .top span.clear {
93      background: #ff9fa8;
94      box-shadow: 0px 4px #ff7c87;
95      color: white;
96  }
97  /* 数字键鼠标移入效果 */
98  .keys span:hover {
99      background: #9c89f6;
100     box-shadow: 0px 4px #6b54d3;
101     color: white;
102 }
103 /* 运算符鼠标移入效果 */
104 .keys span.eval:hover {

```

```

105     background: #abb850;
106     box-shadow: 0px 4px #717a33;
107     color: #ffffff;
108 }
109 /* 清除键鼠标移入效果 */
110 .top span.clear:hover {
111     background: #f68991;
112     box-shadow: 0px 4px #d3545d;
113     color: white;
114 }
115 /*模拟键盘按下效果：当键盘按下时，去除 box-shadow，并且将该按钮下移一个像素*/
116 /*……省略部分容器样式，可参考源代码*/

```

在 CSS 代码中，使用了简单的浏览器重置样式，即清除所有元素的内外边距，并且设置了 `box-sizing` 属性值为 `border-box`。`box-sizing` 是 CSS 3 提供的属性，可自定义盒模型，语法格式如下：

```
01 box-sizing: content-box|border-box|inherit;
```

`box-sizing` 规定了元素带边框的宽高的计算方式，其可取值有：`content-box` | `border-box` | `inherit`。`content-box` 是 CSS 2.1 规定的宽度高度行为，宽度和高度分别应用到元素的内容框，在宽度和高度之外绘制元素的内边距和边框。`border-box` 为元素设定的宽度和高度决定了元素的边框盒。也就是说，为元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。`inherit` 规定应从父元素继承 `box-sizing` 属性的值。这一设置重置了 IE 浏览器与非 IE 浏览器的盒模型的默认显示。

CSS 代码第 12~17 行将背景渐变设置为径向渐变。第 19~30 行使用 `box-shadow`，创造计算器具有 3D 特效。第 36~52 行设置计算器显示区域的样式，并使用内阴影 `inset`，创造显示区的缩进效果。

CSS 代码第 61~80 行设置计算器的操作区域的统一样式。其中，第 77 行代码使用 `user-select: none` 设置键盘的内容不可选。`user-select` 是 CSS 3 中新增的属性，目前只有 Gecko 和 Webkit 支持该属性，包括基本上所有版本的 Firefox/Chrome/Safari，IE 10 也支持该属性，Opera 尚不支持。第 79 行使用 CSS 3 transitions，使得计算器键盘区域的按钮样式在不同状态之间的变化更为平滑。

从第 82 行代码起，分别设置了操作符、计算符、清除等按键的特殊样式，以及不同类型的按键的不同状态的特殊样式。详细见代码注释。

11.2 使用 CSS 3 制作网页播放器

毫无疑问，HTML 5 已经是大势所趋，尤其随着移动互联网的发展，在移动终端上的浏览器不支持 Flash，更促进了 HTML 5 的发展。虽然国内还没有完全普及支持 HTML 5 浏览器，但在各大本土浏览器厂商的努力下，支持 HTML 5 的浏览器在中国浏览器市场的

占有率也在不断增长中。本节制作一个基于 HTML 5 & CSS 3 & JavaScript 技术的视频播放器，使用开源的 HTML 5 音、视频插件 MediaElement。

制作网页播放器，需要添加一个 HTML 5 video 标记来创建一个视频播放器，再添加一些属性将它初始化。HTML 代码如下：

```
01 <head>
02     <meta charset="utf-8">
03     <title>11.2</title>
04     <script src="js/jquery.min.js"></script>
05     <script src="js/mediaelement-and-player.min.js"></script>
06     <link rel="stylesheet" href="css/style.css" media="screen">
07     <meta name="robots" content="noindex,follow" />
08 </head>
09 <body>
10     <video width="640" height="267" poster="media/cars.png">
11         <source src="media/cars.mp4" type="video/mp4">
12     </video>
13 <script>
14     $(document).ready(function() {
15         $('video').mediaelementplayer({
16             alwaysShowControls: false,
17             videoVolume: 'horizontal',
18             features: ['playpause','progress','volume','fullscreen']
19         });
20     });
21 </script>
22 </body>
```

CSS 代码如下：

```
01 /* CSS Reset */
02 .mejs-inner,
03 .mejs-inner div,
04 .mejs-inner a,
05 .mejs-inner span,
06 .mejs-inner button {
07     margin: 0;
08     padding: 0;
09     border: none;
10     outline: none;
11 }
12 /* 设置视频容器样式 */
13 .mejs-container {
14     position: relative;
15     background: #000000;
16 }
17 .mejs-inner {
```



```
18     position: relative;
19     width: inherit;
20     height: inherit;
21 }
22 .mejs-plugin {
23     position: absolute;
24 }
25 .mejs-container-fullscreen .mejs-mediaelement,
26 .mejs-container-fullscreen video,
27 .mejs-embed,
28 .mejs-embed body,
29 .mejs-mediaelement {
30     width: 100%;
31     height: 100%;
32 }
33 .mejs-embed,
34 .mejs-embed body {
35     margin: 0;
36     padding: 0;
37     overflow: hidden;
38 }
39 .mejs-container-fullscreen {
40     position: fixed;
41     left: 0;
42     top: 0;
43     right: 0;
44     bottom: 0;
45     overflow: hidden;
46     z-index: 1000;
47 }
48 .mejs-poster img {
49     display: block;
50 }
51 .mejs-background,
52 .mejs-mediaelement,
53 .mejs-poster,
54 .mejs-overlay {
55     position: absolute;
56     top: 0;
57     left: 0;
58 }
59 .mejs-overlay-play {
60     cursor: pointer;
61 }
62 .mejs-inner .mejs-overlay-button {
```

```

63     position: absolute;
64     top: 50%;
65     left: 50%;
66     width: 50px;
67     height: 50px;
68     margin: -25px 0 0 -25px;
69     background: url(..img/play.png) no-repeat;
70 }
71 /* 设置控制条容器样式 */
72 .mejs-container .mejs-controls {
73     position: absolute;
74     width: 100%;
75     height: 34px;
76     left: 0;
77     bottom: 0;
78     background: rgb(0,0,0); /* IE8- */
79     background: rgba(0,0,0, .7);
80 }
81 /* 设置控制条按钮样式 */
82 .mejs-controls .mejs-button button {
83     display: block;
84     cursor: pointer;
85     width: 16px;
86     height: 16px;
87     background: transparent url(..img/controls.png);
88 }
89 /* 设置播放、暂停按钮样式 */
90 .mejs-controls div.mejs-playpause-button {
91     position: absolute;
92     top: 12px;
93     left: 15px;
94 }
95 .mejs-controls .mejs-play button,
96 .mejs-controls .mejs-pause button {
97     width: 12px;
98     height: 12px;
99     background-position: 0 0;
100 }
101 .mejs-controls .mejs-pause button {
102     background-position: 0 -12px;
103 }
104 /* .....省略部分容器样式，可参考源代码 */

```

此段 CSS 代码展示了播放器基本样式设计。CSS 代码第 1~11 行是 CSS Reset 部分，重置了关键元素的内外边距与边框，均设置为 0。第 13~58 行是播放器的布局设置部分。第

59~70 行是播放按钮的样式。代码第 71~88 行添加视频控制器布局，将它放在视频底部，高度为 34px，再添加一个背景颜色，配合 RGBA 来设置透明度，最后给按钮添加基本样式和图片。

在播放器最终显示效果中，当视频尚未播放时，仅显示默认图片的效果如图 11.2、图 11.3 所示，视频播放时的效果如图 11.4、图 11.5 所示。



图 11.2 CSS 3 播放器视频未播放显示控制条



图 11.3 CSS 3 播放器视频未播放隐藏控制条



图 11.4 CSS 3 播放器视频播放中显示滚动条



图 11.5 CSS 3 播放器视频播放中隐藏滚动条

11.3 不使用 table 的 Form 表单

在 Form 表单中，常常会遇到不同表单项不对齐的情况，例如，当提示文字字数不等的情况下，常常会不对齐，如图 11.6 所示。常用的解决方案是使用 table，强制实现对齐。本节介绍一种不使用 table 的 Form 表单的实现方法。

用户名：	<input type="text"/>
密码：	<input type="password"/>

图 11.6 Form 表单的对齐问题

为了便于控制显示文字与表单的对齐等相关显示，本节提出了一种不使用 table 的实现方法，转而使用无序列表来实现。实例 HTML 代码如下：

```

01 //表单结构，使用无序列表 ul 的列表元素 li 设置表单各项内容
02 <form class="stepnow-fillin" id="J_Form" action="../../../pages/io/post.json">
03     <fieldset class="mod mod-contacter">
04         <header class="hd">填写联系人信息</header>
05         <div class="bd " id="J_ContacterWrapper">
06             <ul class="contacter-info" id="J_Contacter">
07                 <li>
08                     <label class="label" for="J_ContacterName">姓名： </label>
09                     <em class="required">*</em>
10                     <span class="verify-wrap">
11                         <input type="text" class="input-text" id="J_ContacterName" name=
"relationName"/>
12                     </span>
13                 </li>
14                 <li>
15                     <label class="label" for="J_ContacterTel">手机号码： </label>

```

```

16         <em class="required">*</em>
17         <span class="verify-wrap">
18             <input type="text" class="input-text" id="J_ContacterTel" placeholder=
"通知机票出票状态和航班信息" name="relationMobile"/>
19         </span>
20     </li>
21 </li>
22     <label class="label" for="J_ContacterPhone"> 备选号码: </label>
23     <em class="required">&nbsp;</em>
24     <span class="verify-wrap">
25         <input type="text" class="input-text" id="J_ContacterPhone"
placeholder="" name="relationPhone"/>
26     </span>
27 </li>
28 </li>
29     <label class="label" for="J_ContacterEmail"> 电子邮箱: </label>
30     <em class="required">*</em>
31     <span class="verify-wrap">
32         <input type="text" class="input-text" id="J_ContacterEmail"
placeholder="通知机票出票状态和航班信息" name="relationEmail"/>
33     </span>
34 </li>
35 <li class="save-relation-list">
36     <label class="label"></label>
37     <em class="required"></em>
38     <label class="save-relation-label" for="J_NeedSaveRelation">
39         <input type="checkbox" name="needSaveRelation" class=
"save-relation" checked="true" id="J_NeedSaveRelation"/>
40         存为常用联系人
41     </label>
42 </li>
43 </ul>
44 </div>
45 </fieldset>
46 </form>

```

CSS 代码如下:

```

01  /*CSS Reset*/
02  fieldset, img {
03      border: 0;
04  }
05  em {
06      margin: 0;
07      padding: 0;
08      border: 0;

```

```

09     font-size: 100%;
10 }
11 /* 必填项样式 */
12 .required {
13     color: #e90000;
14     width: 20px;
15     display: inline-block;
16     height: 12px;
17     line-height: 12px;
18     overflow: hidden;
19 }
20 /* 提示文字样式 */
21 .label {
22     display: inline-block;
23     *display: inline;
24     *zoom: 1;
25     width: 185px;
26     text-align: right;
27     height: 32px;
28     line-height: 32px;
29 }
30 /* 表单输入框样式 */
31 .input-text {
32     display: inline-block;
33     *display: inline;
34     *zoom: 1;
35     width: 225px;
36     height: 18px;
37     line-height: 18px;
38     overflow: hidden;
39     padding: 6px 3px;
40     border: 1px solid #bababa;
41     box-shadow: 0 0 3px #e6e6e6 inset;
42     border-radius: 3px;
43     outline: 0;
44 }
45 /* 鼠标移入表单样式 */
46 .input-text:hover {
47     border-color: #666666;
48 }
49 /* 表单取得焦点样式 */
50 .input-text:focus {
51     border-color: #0092d2;
52 }
53 /* 下拉列表的样式 */

```

```
54 .input-select {
55     border: 1px solid #bababa;
56     padding: 4px 5px 4px 3px;
57     height: 30px;
58     vertical-align: middle;
59     line-height: 1;
60     font-size: 13px;
61     width: 233px;
62     border-radius: 3px;
63     outline: 0;
64 }
65 /* 下拉列表鼠标移入的样式 */
66 .input-select:hover {
67     border-color: #666666;
68 }
69 /* 下拉列表取得焦点的样式 */
70 .input-select:focus {
71     border-color: #0092d2;
72 }
73 /* 标题样式 */
74 .mod .hd {
75     font-size: 16px;
76     font-weight: bold;
77     color: #404040;
78     font-family: "Microsoft YaHei";
79     height: 16px;
80     line-height: 16px;
81     overflow: hidden;
82     border-bottom: 2px solid #0092d2;
83     padding-bottom: 6px;
84 }
85 /* 表单容器样式 */
86 .mod-contacter {
87     padding: 20px;
88     padding-bottom: 1px;
89     *margin-bottom: -3px;
90 }
91 /* 表单列表样式 */
92 .mod-contacter li {
93     padding: 7px 0;
94     height: 32px;
95     line-height: 32px;
96     overflow: hidden;
97 }
98 .mod-contacter .contacter-info, .mod-contacter .invoice {
```

```

99     padding: 22px 0 18px 0;
100  }
101  /* .....省略部分容器样式，可参考源代码 */

```

不使用 `table` 的表单在结构设计时，选择使用无序列表 `ul` 来定义表单，每个输入框均由 `li` 包裹，并且，提示信息均使用 `label` 进行封装。必填提示符 `*` 号使用 `em` 元素包裹。在 CSS 代码中，使用 CSS Reset，重置了 `fieldset`、`em` 的样式，并为全局必填提示 `required`、文本提示 `label`、文本输入框 `input-text`、下拉列表 `input-select` 等分别定义样式。

本例最终展示效果如图 11.7 所示。

图 11.7 不使用 `table` 的 Form 表单

11.4 可以重复利用的规则

在制作网页的过程中，常常会遇到不同位置的元素需要重复使用相同的样式，也会遇到相同元素在不同结构下被重复定义、但样式又不同的情况，基于这两点原因，首先需要了解 CSS 权重。

CSS 权重决定了 CSS 的哪一条样式规则会被浏览器应用在元素上，当很多的规则被应用到某一个元素上时，权重来决定哪种规则生效和计算优先级。每个选择器都有自己的权重，每条 CSS 样式规则都包含一个权重级别。这个级别是由不同的选择器加权计算的，通过权重，不同的样式最终会作用到对应的元素上。如果两个选择器同时作用到一个元素上，权重较高的一个样式规则会被应用于目标元素上。例如：

```

//CSS 代码
01  a {
02    border-bottom: 0px;
03    color: #5DB0E6;
04  }
05  a:focus {
06    outline: 1px dotted #eee;
07  }
08  a:active {
09    outline: 0px;

```



```
10 }
11 a:hover {
12     color: #7bc4f4;
13 }
14 body#jq-interior {
15     background-image: url(http://static.jquery.com/files/rocker/images/bg-interior-tile-drk.jpg);
16 }
```

以上样式表是由一个个 CSS 层叠样式规则组成,而每一个样式规则又可以分为两部分:选择符与声明。选择符能针对特定元素进行设置。CSS 又叫层叠样式表,基本上被选中的元素的子元素能继承它的样式,但其子元素如果设置了样式,也能覆盖样式。CSS 选择器的权重见表 11.1。

表 11.1 选择器权重

选择器	表达式	说明	权重
ID选择器	#aaa		100
类选择器	.aaa		10
标签选择器	h1	元素的tagName	1
属性选择器	[href]		10
相邻选择器	selector + selector	拆分为两个选择器再计算	
兄长选择器	selector ~ selector	拆分为两个选择器再计算	
亲子选择器	selector > selector	拆分为两个选择器再计算	
后代选择器	selector selector	拆分为两个选择器再计算	
通配符选择器	*		0
各种伪类选择器	如:link, :visited, :hover, :active, :target, :root, :not等		10
各种伪元素	如::first-letter,::first-line,::after,::before,::selection		1

根据表 11.1 所示的权重,可以计算出以下 CSS 代码的选择器权重,具体如下:

```
//CSS 代码
01 *          {} /* a=0 b=0 c=0 d=0 ->权重= 0,0,0,0 */
02 li         {} /* a=0 b=0 c=0 d=1 ->权重= 0,0,0,1 */
03 li:first-line {} /* a=0 b=0 c=0 d=2 ->权重= 0,0,0,2 */
04 ul li      {} /* a=0 b=0 c=0 d=2 ->权重= 0,0,0,2 */
05 ul ol+li   {} /* a=0 b=0 c=0 d=3 ->权重= 0,0,0,3 */
06 h1 + *[rel=up] {} /* a=0 b=0 c=1 d=1 ->权重= 0,0,1,1 */
07 ul ol li.red {} /* a=0 b=0 c=1 d=3 ->权重= 0,0,1,3 */
08 li.red.level {} /* a=0 b=0 c=2 d=1 ->权重= 0,0,2,1 */
09 #x34y      {} /* a=0 b=1 c=0 d=0 ->权重= 0,1,0,0 */
10 style=""    {} /* a=1 b=0 c=0 d=0 ->权重= 1,0,0,0 */
```

因此,当需要重复利用 CSS 规则时,需要综合考虑 CSS 权重。

11.5 在同一元素上使用多种类

在第 11.4 节中，介绍了 CSS 权重的计算方法，根据权重的计算结果，可以为元素设计特定的样式，已达到代码的复用。为了达到代码的复用，本节介绍在同一元素上使用多种 class 类，以达到代码复用的效果，减少 CSS 代码量。实例代码如下：

```
//HTML 代码
<p class="red bold pb10 m10">同一元素使用两种类</p>

//CSS 代码
01  .red {
02      color: red;
03  }
04  .bold {
05      font-weight: bold;
06  }
07  /* padding */
08  .pt5 {
09      padding-top: 5px;
10  }
11  .pb5 {
12      padding-bottom: 5px;
13  }
14  .pl5 {
15      padding-left: 5px;
16  }
17  /*
18  * .....
19  * 省略部分容器样式，如.pr5/.pt10 系列/.pt15 系列等
20  * 可参考源代码
21  */
22  /* margin */
23  .mc {
24      margin: auto;
25  }
26  .mg5 {
27      margin: 5px;
28  }
29  .mt5 {
30      margin-top: 5px;
31  }
32  /*
33  * .....
34  * 省略部分容器样式，如.mr5/.mt10 系列/.mt15 系列等
35  * 可参考源代码
```

36 */

此段 CSS 代码分别定义了一些常用的字体、字号、内边距、外边距等样式设置，虽然单纯定义这些代码较为烦琐，但是，在构建大型网站的时候，抽离这些公共样式，并在全站引用该样式文件，不仅减少了重复代码的书写，而且，便于网站升级或是二次开发。

11.6 CSS 块引用模板

块引用 `blockquote` 元素在符合 W3C 标准浏览器上均会在引文前呈现一个大的双引号，并且 IE 浏览器还会在块引用处显示一个较粗的左边框与浅灰色背景色。与其他块元素不同的是 `blockquote` 不需要其他块级嵌套元素，因此，`blockquote` 将段落呈现为行内元素样式。本节介绍一种 CSS 块引用模板的实现方法，最终展示效果如图 11.8 所示。

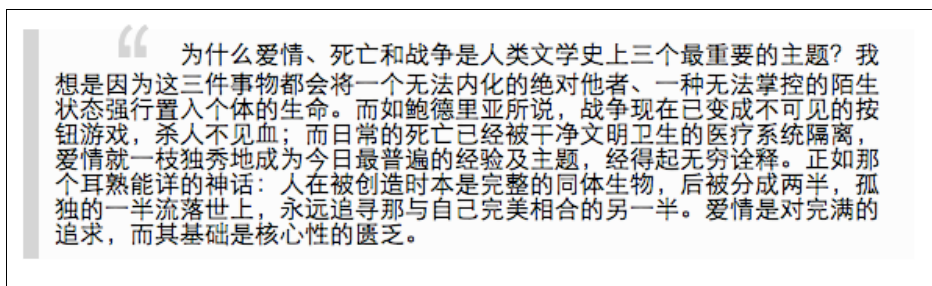


图 11.8 块引用效果

实例代码如下：

```
//CSS 代码
01  blockquote {
02      background: #f9f9f9;
03      border-left: 10px solid #ccc;
04      -webkit-border-radius: 3px;
05      -moz-border-radius: 3px;
06      border-radius: 3px;
07      font-style: italic;
08      margin: 1.5em 10px;
09      padding: 0.5em 10px;
10      quotes: "\201C""\201D""\2018""\2019";
11  }
12  blockquote:before {
13      color: #ccc;
14      content: open-quote;
15      font-size: 4em;
16      line-height: 0.1em;
17      margin-right: 0.25em;
18      vertical-align: -0.4em;
```

```

19  }
20  blockquote p {
21      display: inline;
22  }
//HTML 代码
23  <div class="example">
24      <blockquote>为什么爱情、死亡和战争是人类文学史上三个最重要的主题？……//省略部
        分文字。爱情是对完满的追求，而其基础是核心性的匮乏。<p></p></blockquote>
25  </div>

```

此段 CSS 代码通过修改 `blockquote` 标签样式，让文章的引用段落显示的漂亮一些，应用以上代码，使用 `blockquote` 标签引用一段文字，将可以呈现统一的引用效果。

11.7 花式 CSS 3 Pull-引文

Pull-引文（Pull-quotes）与块引用（blockquotes）不同，这种引文通常显示在博客或者新闻文章的一边。这些引文经常从文章中引用文本，所以它们和块引用显示的稍许不同。这些默认类具有一些基础的属性，带有 3 个可供选择的独特的字体类型。实例代码如下：

```

//CSS 代码
01  .has-pullquote:before {
02      /* 重置矩形 */
03      padding: 0;
04      border: none;
05      /* Content */
06      content: attr(data-pullquote);
07      /* 居右 基于模块化的引文 */
08      float: right;
09      width: 320px;
10      margin: 12px -140px 24px 36px;
11      /* Baseline 矫正 */
12      position: relative;
13      top: 5px;
14      /* 排版 (30px line-height = 25% incremental leading) */
15      font-size: 23px;
16      line-height: 30px;
17  }
18  .pullquote-adelle:before {
19      font-family: "adelle-1", "adelle-2";
20      font-weight: 100;
21      top: 10px !important;
22  }
23  .pullquote-helvetica:before {
24      font-family: "Helvetica Neue", Arial, sans-serif;

```

```

25     font-weight: bold;
26     top: 7px !important;
27 }
28 .pullquote-facit:before {
29     font-family: "facitweb-1", "facitweb-2", Helvetica, Arial, sans-serif;
30     font-weight: bold;
31     top: 7px !important;
32 }
//HTML 代码
33 <h3>Example</h3>
34 <div class="example has-pullquote">
35     <blockquote>为什么爱情、死亡和战争是人类文学史上三个最重要的主题? .....//省略部
        分文字。爱情是对完满的追求, 而其基础是核心性的匮乏。<p></p></blockquote>
36 </div>

```

花式 Pull-引文见 CSS 代码部分, 第 3 行重置了内边距, 并且在第 8 行设置向右浮动, 第 12 行设置相对定位, 第 13 行设置了距离顶部的距离 `top: 5px`, 并定义了文字排版样式, 如字体大小、行高等。

11.8 一般媒体查询

固定宽度设计的静态网站在用户友好性上, 比不上使用灵活的响应式设计的网站, 该设计可以实现根据屏幕大小进行上扩和下扩。利用响应式设计, 无论用户使用哪种类型的设备或屏幕来访问网站, 都可以为其呈现一个可用的界面(如图 11.9)。用户屏幕将随着新的智能手机和平板电脑的问世而快速演变, 利用响应式设计就可以响应各种屏幕大小。实现响应式设计的主要途径是使用 CSS 的媒体查询。本节介绍了如何将媒体查询用于桌面网站、移动电话和平板电脑。



图 11.9 媒体查询

首先媒体查询的先决条件是运行于支持 CSS 媒体查询的网页浏览器中, 这些浏览器包括 Mozilla Firefox、Apple Safari、Google Chrome 和 Opera。从 CSS 2 开始, 就可以通过媒体类型在 CSS 中获得媒体支持。响应式设计可根据所显示的屏幕大小而改变, 它呈现的每个屏幕看起来并不相同。按照可用的屏幕属性, 响应式设计提供了 UI 的最佳效果。通过媒

体查询的设置，程序编写的 CSS 可自动将设计更改为提供不同屏幕大小的最佳 UI 体验。以下代码展示了一个实例：

```
//HTML 代码
<link rel="stylesheet" type="text/css" href="site.css" media="screen" />
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

从上面代码中可以看出，`media` 属性定义了应该用于指定每种媒体类型的样式表，其中 `screen` 适用于计算机彩色屏幕，`print` 适用于打印预览模式下查看的内容或者打印机打印的内容。媒体查询作为 CSS 3 规范的一部分，可以扩展媒体类型函数，并允许在样式表中使用更精确的显示规则。媒体查询是用 `true` 或 `false` 来选择性加载：如果为 `true`，则继续使用样式表，如果为 `false`，则不能使用样式表。这种简单逻辑通过表达式变得更加强大，设计时能够更灵活地对特定的设计场景使用自定义的显示规则。

媒体查询包含一个媒体类型，后面可以加上一个或多个检查特定条件的表达式。CSS 样式表中的媒体查询语法：

```
//CSS 代码
@media all and (min-width: 800px) {
    /* CSS 代码 */
}
```

以上代码定义了所有最小水平屏幕宽度为 800 像素的屏幕（屏幕和打印等）都应使用的 CSS 规则，该规则在实例第 2 行的地方进行定义。该媒体查询 `@media all` 是媒体类型，也就是说，可将此 CSS 应用于所有媒体类型。`min-width:800px` 是包含媒体查询的表达式，如果浏览器的最小宽度为 800 像素，则浏览器只运用其中的 CSS 规则。

可以省略关键词 `all` 和 `and`。在将某个媒体查询应用于所有媒体类型时，可以省略 `all`，`and` 也是可选的。使用简写语法重新编写媒体查询：

```
//CSS 代码
@media (min-width: 800px) {
    /* CSS 代码 */
}
```

媒体查询也可以包含复杂表达式。例如，如果想要创建一个仅在最小宽度为 800 像素且最大宽度为 1024 像素时应用的样式，则需要按照复杂的规则来设计：

```
//CSS 代码
@media (min-width:800px) and (max-width:1024px){
    /* CSS 代码 */
}
```

还可以使用多个 `and` 表达式，如需要增加其他条件来检查特定的屏幕方向，只需添加另一个 `and` 关键词，加上 `orientation` 媒体查询：

```
//CSS 代码
@media (min-width:800px) and (max-width:1024px) and (orientation:portrait){
    /* CSS 代码 */
}
```

类似地，与 `and` 相似，可以使用 `or` 表达式。与 `and` 一样，多个条件组合在一起会构成复杂表达式。使用 `or` 时，如果其中有一个条件为 `true`，那么整个表达式或分离的两个条件

都会为 true，如：

```
//CSS 代码
@media (min-width:800px) or (orientation:portrait){
    /* CSS 代码 */
}
```

另一个保存在词库中的媒体查询关键词是 not。位于媒体查询的开始处，not 会忽略结果。换句话说，如果该查询本来在没有 not 关键词的情况下为 true，那么现在它将为 false。如：

```
//CSS 代码
@media (not min-width:800px){
    /* CSS 代码 */
}
```

not 表达式定义了当最小宽度不是 800 像素时会应用的 CSS 规则。

以上这些实例只是将像素作为媒体查询中的测量单位，但是测量单位并不仅限于像素，也可以使用任何有效的 CSS 测量单位，比如厘米（cm）、英寸（in）、毫米（mm）等。

总之，使用 CSS 媒体查询技术可以轻松地为特定屏幕大小创建可靠的用户体验，不论用户将会使用何种浏览器或设备来访问。媒体查询技术是响应式设计的中中之重，响应式设计是一个新兴移动 Web 设计和开发实践，使得用户在平板电脑、手机或电子阅读器上访问网站有良好的体验。

11.9 CSS 3 背景梯度

CSS 3 梯度即 CSS 3 Gradient，为实现渐变提供了一种实现方法。CSS 3 Gradient 分为 linear-gradient（线性渐变）和 radial-gradient（径向渐变），这在 4.1 节已经都详细的介绍过了，本节不再赘述。本例用 CSS 3 Gradient 实现一个背景梯度，效果如图 11.10 所示。



图 11.10 CSS 3 背景梯度模板

实例代码如下：

```
//HTML 代码
<div class="colorbox wrap">
```

```

</div>
//CSS 代码
01 .wrap {
02     width: 200px;
03     height: 200px;
04     border: 1px solid #dfdfdf;
05 }
06 .colorbox {
07     background: #629721;
08     /* 背景颜色渐变 */
09     background-image: -webkit-gradient(linear, left top, left bottom, from(#83b842), to(#629721));
10     background-image: -webkit-linear-gradient(top, #83b842, #629721);
11     background-image: -moz-linear-gradient(top, #83b842, #629721);
12     background-image: -ms-linear-gradient(top, #83b842, #629721);
13     background-image: -o-linear-gradient(top, #83b842, #629721);
14     background-image: linear-gradient(top, #83b842, #629721);
15 }

```

此段代码展示了添加背景梯度的模板方法，当使用时，只需将渐变的起止颜色替换为所需的颜色即可。

11.10 CSS 日历显示效果

本节介绍一种 CSS 3 日历的实现方法。按月显示的日历由上一月、当前月和下一月组成，因此在设计日历的 HTML 结构代码中，需要设置出上一月、当前月和下一月的结构代码，以便设置样式。在日历样式设置中，需分别设置出上一月、当前月和下一月的样式代码，包括日历数字、背景等的显示。CSS 日历的重点在于 HTML 结构，良好的 HTML 结构能实现较为便捷的样式控制。该日历的最终展示效果如图 11.11 所示。



图 11.11 CSS 3 日历

实例 HTML 代码如下:

```
01 <h1>&larr; July &rarr;</h1>
02 <ol class="calendar" start="6">
03     <li id="lastmonth">
04         <ul start="29">
05             <li>29</li>
06             <li>30</li>
07         </ul>
08     </li>
09     <li id="thismonth">
10         <ul>
11             <li>1</li>
12             <li>2</li>
13             <li>3</li>
14             <li class="holiday">4
15                 <p>
16                     <a href="http://www.865171.cn/css-code/">CSS 代码实例</a>
17                 </p>
18             </li>
19             <li>5</li>
20             <li>6</li>
21             <li>7</li>
22             <li>8</li>
23             <li>9
24                 <p>
25                     <a href="http://www.865171.cn/css-code/">CSS 代码实例</a>
26                 </p>
27             </li>
28             <li>10</li>
29             <li>11</li>
30             <li>12</li>
31             <li>13</li>
32             <li>14</li>
33             <li>15</li>
34             <li>16</li>
35             <li>17
36                 <p>
37                     <a href="http://www.865171.cn/css-code/">CSS 代码实例</a>
38                 </p>
39             </li>
40             <li>18</li>
41             <li>19</li>
42             <li>20</li>
43             <li>21</li>
```

```

44         <li>22</li>
45         <li>23</li>
46         <li>24</li>
47         <li>25</li>
48         <li>26</li>
49         <li>27</li>
50         <li>28</li>
51         <li>29</li>
52         <li>30</li>
53         <li>31</li>
54     </ul>
55 </li>
56 <li id="nextmonth">
57     <ul>
58         <li>1</li>
59         <li>2</li>
60     </ul>
61 </li>
62 </ol>

```

CSS 代码如下：

```

01 /* CSS Reset */
02 * {
03     margin: 0;
04     padding: 0;
05 }
06 body {
07     font: 1em/1.4 Verdana, Arial, Helvetica, sans-serif;
08     background: url(images/bg.jpg) top center no-repeat #545454;
09 }
10 body * {
11     display: inline;
12 }
13 ol.calendar {
14     width: 52em;
15     margin: 0 auto;
16     display: block;
17     min-height: 200px;
18     background: url(images/tl.png) top left no-repeat;
19     padding: 12px 0 0 20px;
20 }
21 li {
22     list-style: none;
23 }
24 p.link {

```

```
25     text-align: center;
26     display: block;
27 }
28 h1 {
29     display: block;
30     width: 200px;
31     height: 76px;
32     background: url(images/july.png);
33     text-indent: -9999px;
34     margin: 15px auto;
35 }
36 /* 天的样式 */
37 li li {
38     width: 6em;
39     height: 6em;
40     float: left;
41     margin: .2em;
42     padding: .2em;
43     overflow: auto;
44     background: url(images/day-bg.png) bottom right no-repeat;
45 }
46 /* 日历内容样式 */
47 li li p {
48     font-size: .7em;
49     display: block;
50 }
51 li li ol {
52     width: auto;
53 }
54 li li ul li,
55 li li ol li {
56     font-size: .7em;
57     display: block;
58     height: auto;
59     width: auto;
60     background: none;
61     margin: 0;
62     padding: .2em 0;
63     float: none;
64 }
65 /* 假期样式 */
66 li li.holiday {
67 }
68 /* 月的样式 */
69 li#lastmonth li,
```

```

70 li#nextmonth li {
71     background: url(images/day-bg-inactive.png);
72 }

```

该日历在设计 HTML 结构代码时，采用了有序列表与无序列表嵌套的结构，整体日历为一个有序列表 `ol`，上一个月与下一个月的日期分别在该 `ol` 内的一个列表 `li` 里，并分别用无序列表 `ul` 包裹。当前月的日期在整体日历 `ol` 内部的列表 `li` 里，且日历上的记事也在 `li` 内部包裹。

因此，在实现 CSS 代码时，分别对整体日历、上一月、当前月、下一月做修饰，采取不同的颜色加以区分。每一天的背景由图片实现。

11.11 字符编码

几乎所有的代码编写都涉及字符编码的问题，只要有中文的地方，总是会遇到各种编码问题，并且这种问题还非常难缠，因为代码都是用英文开发的，基本很少考虑其他语种编码问题，从而导致了后续涉及其他种语言时，出现 N 多编码问题。

首先，要熟悉文件的存储方式。文件都有自己的存储格式，比如最常见的 `txt`、`cpp`、`h`、`c`、`xml`、`png`、`rmvb` 等各种格式，此外还有自定义格式。这些文件不论是什么格式，都是在计算机硬盘里的二进制格存储，对应不同文件格式，有不同的软件解析。因此，还需要了解文本文件解析。

二进制文件对应于存储文件，文本文件才是对应于人类可以阅读的文本，如何从二进制转换为文本文件呢？起初由于计算机在美国发明，自然考虑的是英语如何表示，英语字母总共 26 个，再加上一些特殊字符，共 128 个字符，使用 7 byte 即可表示出来，即 ASCII 编码，其对应关系很简单，一个字符对应一个 byte。但是其他非英语国家的文字远远超过 ASCII 码的范围，这时候需要统一字符编码，不同国家使用自己不同的编码方式，例如我国的 `gb2312`。倘若每个国家一种编码方式，在通用性上就会出问题。Unicode 编码是希望将编码统一的一种编码方式，因此其规定了每个字符对应的 Unicode 码。

如何解决编码问题呢？可以借助于编辑器来解决。例如在 `vim` 编辑器的配置文件中去掉配置命令：

```
set encoding=utf-8 set no bomb
```

添加命令：

```
set encoding=utf-8 set bomb
```

此外还可以使用 Notepad++ 编辑器自带的编码转换功能来解决。

在开发 HTML 文件的 `<head>` 标签中，添加：

```
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

或：

```
<meta charset="utf-8" />
```

在独立的 CSS 文件中，添加：

```
@charset "utf-8";
```

11.12 手机 APP 使用的简洁注册页面

本节介绍一种手机 APP 使用的简洁注册页面。由于在手机端用户输入信息较为麻烦，因此手机 APP 的注册界面较之于 PC 端的注册页面要简洁许多，仅填写重要信息即可，效果如图 11.12 所示。

图 11.12 APP 注册页面

实例 HTML 代码如下：

```

01  <!--注册表标题 -->
02  <h1 class="register-title">Welcome</h1>
03  <!--注册表单 form -->
04  <form class="register">
05  <!--注册表—— 性别 -->
06  <div class="register-switch">
07  <input type="radio" name="sex" value="F" id="sex_f" class="register-switch-input"
checked="">
08  <label for="sex_f" class="register-switch-label">Female</label>
09  <input type="radio" name="sex" value="M" id="sex_m" class="register-switch-input">
10  <label for="sex_m" class="register-switch-label">Male</label>
11  </div>
12  <!--注册表—— 电子邮件 -->
13  <input type="email" class="register-input" placeholder="Email address">
14  <!--注册表—— 密码 -->
15  <input type="password" class="register-input" placeholder="Password">

```

```

16      <!--注册表—— 提交按钮 -->
17      <input type="submit" value="Create Account" class="register-button">
18  </form>
19  <!--注册表—— 关于 -->
20  <div class="about">
21      <p class="about-links">
22          <a href="https://github.com/yinqiao/supercss" target="_parent">View Article</a>
23          <a href="https://github.com/yinqiao/supercss" target="_parent">Download</a>
24      </p>
25      <p class="about-author">
26          © 2014
27          <a href="http://yinqiao.github.io/2014/03/16/supercss/">超实用代码段</a>
28      </p>
29  </div>

```

CSS 代码如下:

```

01  /*设置字体及颜色*/
02  body {
03      font: 14px/20px 'Helvetica Neue', Helvetica, Arial, sans-serif;
04      color: #404040;
05      background: #2d4259;
06  }
07  /*注册页标题样式*/
08  .register-title {
09      width: 270px;
10      line-height: 43px;
11      margin: 50px auto 20px;
12      font-size: 19px;
13      font-weight: 500;
14      color: white;
15      color: rgba(255, 255, 255, 0.95);
16      text-align: center;
17      text-shadow: 0 1px rgba(0, 0, 0, 0.3);
18      background: #d7604b;
19      border-radius: 3px;
20      background-image: -webkit-linear-gradient(top, #dc745e, #d45742);
21      background-image: -moz-linear-gradient(top, #dc745e, #d45742);
22      background-image: -o-linear-gradient(top, #dc745e, #d45742);
23      background-image: linear-gradient(to bottom, #dc745e, #d45742);
24      -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.1), inset 0 0 0 1px rgba(255, 255, 255, 0.05), 0 0 1px 1px rgba(0, 0, 0, 0.1), 0 1px 3px rgba(0, 0, 0, 0.3);
25      box-shadow: inset 0 1px rgba(255, 255, 255, 0.1), inset 0 0 0 1px rgba(255, 255, 255, 0.05), 0 0 1px 1px rgba(0, 0, 0, 0.1), 0 1px 3px rgba(0, 0, 0, 0.3);
26  }
27  /*注册 form 样式*/

```

```
28 .register {
29     margin: 0 auto;
30     width: 230px;
31     padding: 20px;
32     background: #f4f4f4;
33     border-radius: 3px;
34     -webkit-box-shadow: 0 0 1px 1px rgba(0, 0, 0, 0.1), 0 1px 3px rgba(0, 0, 0, 0.3);
35     box-shadow: 0 0 1px 1px rgba(0, 0, 0, 0.1), 0 1px 3px rgba(0, 0, 0, 0.3);
36 }
37 /*注册表单样式*/
38 input {
39     font-family: inherit;
40     font-size: inherit;
41     color: inherit;
42     -webkit-box-sizing: border-box;
43     -moz-box-sizing: border-box;
44     box-sizing: border-box;
45 }
46 /*注册输入框样式*/
47 .register-input {
48     display: block;
49     width: 100%;
50     height: 38px;
51     margin-top: 2px;
52     font-weight: 500;
53     background: none;
54     border: 0;
55     border-bottom: 1px solid #d8d8d8;
56 }
57 /*注册输入框获得焦点样式*/
58 .register-input:focus {
59     border-color: #1e9ce6;
60     outline: 0;
61 }
62 /*注册提交按钮样式*/
63 .register-button {
64     display: block;
65     width: 100%;
66     height: 42px;
67     margin-top: 25px;
68     font-size: 16px;
69     font-weight: bold;
70     color: #494d59;
71     text-align: center;
72     text-shadow: 0 1px rgba(255, 255, 255, 0.5);
```

```

73     background: #fcfcfc;
74     border: 1px solid;
75     /*注册提交按钮边框*/
76     border-color: #d8d8d8 #d1d1d1 #c3c3c3;
77     border-radius: 2px;
78     cursor: pointer;
79     /*注册提交按钮背景颜色渐变*/
80     background-image: -webkit-linear-gradient(top, #fefefe, #eeeeee);
81     background-image: -moz-linear-gradient(top, #fefefe, #eeeeee);
82     background-image: -o-linear-gradient(top, #fefefe, #eeeeee);
83     background-image: linear-gradient(to bottom, #fefefe, #eeeeee);
84     -webkit-box-shadow: inset 0 -1px rgba(0, 0, 0, 0.03), 0 1px rgba(0, 0, 0, 0.04);
85     /*注册提交按钮盒阴影*/
86     box-shadow: inset 0 -1px rgba(0, 0, 0, 0.03), 0 1px rgba(0, 0, 0, 0.04);
87 }
88 /*注册按钮激活样式*/
89 .register-button:active {
90     background: #eee;
91     /*注册提交按钮激活时的边框*/
92     border-color: #c3c3c3 #d1d1d1 #d8d8d8;
93     /*注册提交按钮激活时的阴影*/
94     background-image: -webkit-linear-gradient(top, #eeeeee, #fcfcfc);
95     background-image: -moz-linear-gradient(top, #eeeeee, #fcfcfc);
96     background-image: -o-linear-gradient(top, #eeeeee, #fcfcfc);
97     background-image: linear-gradient(to bottom, #eeeeee, #fcfcfc);
98     /*注册提交按钮激活时的盒阴影*/
99     -webkit-box-shadow: inset 0 1px rgba(0, 0, 0, 0.03);
100    box-shadow: inset 0 1px rgba(0, 0, 0, 0.03);
101 }
102 .register-button:focus {
103     outline: 0;
104 }
105 /*注册切换按钮*/
106 .register-switch {
107     height: 32px;
108     margin-bottom: 15px;
109     padding: 4px;
110     background: #6db244;
111     border-radius: 2px;
112     background-image: -webkit-linear-gradient(top, #60a83a, #7dbe52);
113     background-image: -moz-linear-gradient(top, #60a83a, #7dbe52);
114     background-image: -o-linear-gradient(top, #60a83a, #7dbe52);
115     background-image: linear-gradient(to bottom, #60a83a, #7dbe52);
116     -webkit-box-shadow: inset 0 1px rgba(0, 0, 0, 0.05), inset 1px 0 rgba(0, 0, 0, 0.02), inset
-1px 0 rgba(0, 0, 0, 0.02);

```



```

117     box-shadow: inset 0 1px rgba(0, 0, 0, 0.05), inset 1px 0 rgba(0, 0, 0, 0.02), inset -1px 0
        rgba(0, 0, 0, 0.02);
118 }

```

代码首先设置全局字体及颜色，然后再设置注册表的各个部分元素的样式。第 8~26 行设置注册表标题的样式，使用 `text-shadow` 属性设置文字阴影，`text-shadow` 是较为常用的定义方法；使用 `linear-gradient` 设置背景的渐变特效。代码第 42~44 行定义 `box-sizing` 为 `border-box`，这可令浏览器呈现出带有指定宽度和高度的框，并把边框和内边距放入框中，即 `input` 的样式在各个浏览器的表现一致，随后再定义注册输入框样式 `register-input` 的样式。第 62~87 行定义注册提交按钮样式 `register-button` 的样式，使用 `text-shadow` 定义文字阴影，`box-shadow` 定义按钮内阴影，`linear-gradient` 定义按钮背景颜色渐变。代码第 106~118 行使用类似的方法定义注册切换按钮（即 `register-switch`）的样式。

11.13 手机简洁价目表

本节介绍一种手机简洁价目表的设计与实现方法。价目列表主要用于展示商品的不同价格及相应的服务，效果如图 11.13 所示。



图 11.13 APP 简洁价目列表

实例 HTML 代码如下：

```

01  <!-- 价目容器 -->
02  <div class="plans">
03      <!-- 商品容器 1 -->
04      <div class="plan">
05          <!-- 商品价格标题 -->
06          <h3 class="plan-title">起始价格</h3>
07          <!-- 商品价格 -->
08          <p class="plan-price">¥ 19 <span class="plan-unit">每月</span></p>
09          <!-- 商品详情 -->
10          <ul class="plan-features">

```

```

11         <li class="plan-feature">2 <span class="plan-feature-name">网站</span></li>
12         <li class="plan-feature">5<span class="plan-feature-unit">GB</span> <span
class="plan-feature-name">存储</span></li>
13         <li class="plan-feature">3 <span class="plan-feature-name">用户</span></li>
14     </ul>
15     <!--商品用户操作按钮 -->
16     <a href="#" class="plan-button">选择</a>
17 </div>
18 <!--商品容器—— 高亮显示当前选中商品 -->
19 <div class="plan plan-highlight">
20     <p class="plan-recommended">推荐</p>
21     <!--当前选中商品价格标题 -->
22     <h3 class="plan-title">Team</h3>
23     <!--当前选中商品价格 -->
24     <p class="plan-price">¥ 49 <span class="plan-unit">每月</span></p>
25     <!--当前选中商品详情 -->
26     <ul class="plan-features">
27         <li class="plan-feature">5 <span class="plan-feature-name">网站</span></li>
28         <li class="plan-feature">20<span class="plan-feature-unit">GB</span> <span
class="plan-feature-name">存储</span></li>
29         <li class="plan-feature">10 <span class="plan-feature-name">用户</span></li>
30     </ul>
31     <!--当前选中商品-用户操作按钮 -->
32     <a href="#" class="plan-button">选择</a>
33 </div>
34 <div class="plan">
35     <h3 class="plan-title">中级</h3>
36     <p class="plan-price">¥ 99 <span class="plan-unit">每月</span></p>
37     <ul class="plan-features">
38         <li class="plan-feature">20 <span class="plan-feature-name">网站</span></li>
39         <li class="plan-feature">50<span class="plan-feature-unit">GB</span> <span
class="plan-feature-name">存储</span></li>
40         <li class="plan-feature">25 <span class="plan-feature-name">用户</span></li>
41     </ul>
42     <a href="#" class="plan-button">选择</a>
43 </div>
44 <div class="plan">
45     <h3 class="plan-title">企业级</h3>
46     <p class="plan-price">¥ 249 <span class="plan-unit">每月</span></p>
47     <ul class="plan-features">
48         <li class="plan-feature">∞ <span class="plan-feature-name">网站</span></li>
49         <li class="plan-feature">200<span class="plan-feature-unit">GB</span> <span
class="plan-feature-name">存储</span></li>
50         <li class="plan-feature">∞ <span class="plan-feature-name">用户</span></li>
51     </ul>

```

```

52     <a href="#" class="plan-button">选择</a>
53   </div>
54 </div>

```

CSS 代码如下:

```

01  /*字体及颜色设置*/
02  body {
03    font: 13px/20px 'Helvetica Neue', Helvetica, Arial, sans-serif;
04    color: #404040;
05    background: #eeebe4;
06  }
07  /*容器*/
08  .plans {
09    width: 836px;
10    margin: 50px auto;
11    overflow: hidden;
12  }
13  /*单个价目样式*/
14  .plan {
15    float: left;
16    width: 150px;
17    margin: 20px 2px;
18    padding: 15px 25px;
19    text-align: center;
20    background: white;
21    background-clip: padding-box;
22    border: 2px solid #e5ded6;
23    border-color: rgba(0, 0, 0, 0.1);
24    border-radius: 5px;
25  }
26  /*价目标题样式*/
27  .plan-title {
28    margin-bottom: 12px;
29    font-size: 24px;
30    color: #36bce6;
31  }
32  /*价格*/
33  .plan-price {
34    margin-bottom: 20px;
35    line-height: 1;
36    font-size: 28px;
37    font-weight: bold;
38    color: #fd935a;
39  }
40  /*单位*/

```

```

41 .plan-unit {
42     display: block;
43     margin-top: 5px;
44     font-size: 13px;
45     font-weight: normal;
46     color: #aaa;
47 }
48 /*特色样式*/
49 .plan-features {
50     width: 120px;
51     margin: 20px auto 15px;
52     padding: 15px 0 0 15px;
53     border-top: 1px solid #e5ded6;
54     text-align: left;
55 }
56 /*按钮样式*/
57 .plan-button {
58     position: relative;
59     display: block;
60     line-height: 40px;
61     font-size: 16px;
62     font-weight: 500;
63     color: white;
64     text-align: center;
65     text-decoration: none;
66     text-shadow: 0 1px rgba(0, 0, 0, 0.1);
67     background: #fd935c;
68     border-bottom: 2px solid #cf7e3b;
69     border-color: rgba(0, 0, 0, 0.15);
70     border-radius: 4px;
71 }
72 /*按钮激活时的样式*/
73 .plan-button:active {
74     top: 2px;
75     margin-bottom: 2px;
76     border-bottom: 0;
77 }
78 /*推荐按钮样式*/
79 .plan-recommended {
80     width: 160px;
81     margin: -15px auto 15px;
82     padding-bottom: 2px;
83     line-height: 22px;
84     font-size: 14px;
85     font-weight: bold;

```

```

86     color: white;
87     text-shadow: 0 1px rgba(0, 0, 0, 0.05);
88     background: #37bbe6;
89     border-radius: 0 0 4px 4px;
90 }

```

价目列表是商品不同档次的价格列表的展示，其中突出显示推荐价格标准。因此，在 HTML 结构代码设计过程中，并列呈现各个档次的价目元素，价目元素内部再依次陈列详细信息。在 CSS 样式实现的过程中，首先设置全局字体及颜色，再设计单个价目样式，包括价目标题样式、价格、价格单位、按钮、推荐按钮等样式。

11.14 手机简洁任务表

随着用户使用手机的时间越来越多，用户很多行为都改在手机端完成。因此，基于手机的任务管理变得越来越重要，一款简洁、方便的任务表管理对用户来说具有重大意义。本节介绍手机版简洁任务表的设计与实现，效果如图 11.14 所示。

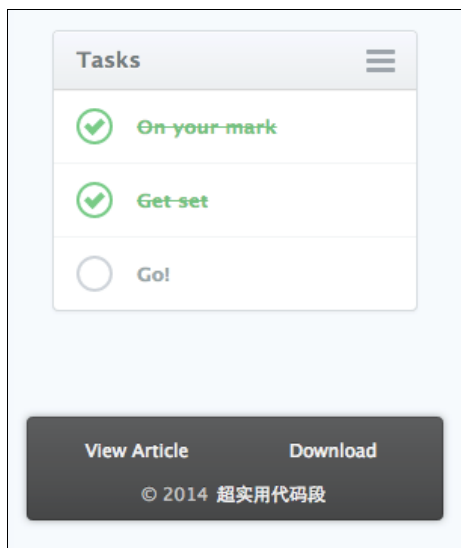


图 11.14 简洁任务列表

实例 HTML 代码如下：

```

01  <!--任务列表容器 -->
02  <section class="tasks">
03      <!--任务标题 -->
04      <header class="tasks-header">
05          <h2 class="tasks-title">Tasks</h2>
06          <a href="index.html" class="tasks-lists">Lists</a>
07      </header>
08      <!--任务列表 -->

```

```

09     <fieldset class="tasks-list">
10     <!--任务详情 -->
11         <label class="tasks-list-item">
12             <input type="checkbox" name="task_1" value="1" class="tasks-list-cb" checked="">
13             <span class="tasks-list-mark"></span>
14             <span class="tasks-list-desc">On your mark</span>
15         </label>
16         <label class="tasks-list-item">
17             <input type="checkbox" name="task_2" value="1" class="tasks-list-cb" checked="">
18             <span class="tasks-list-mark"></span>
19             <span class="tasks-list-desc">Get set</span>
20         </label>
21         <label class="tasks-list-item">
22             <input type="checkbox" name="task_3" value="1" class="tasks-list-cb">
23             <span class="tasks-list-mark"></span>
24             <span class="tasks-list-desc">Go!</span>
25         </label>
26     </fieldset>
27 </section>

```

CSS 代码如下:

```

01  /*设置字体颜色*/
02  body {
03      font: 13px/20px 'Lucida Grande', Verdana, sans-serif;
04      color: #404040;
05      background: #f2f8fc;
06  }
07  /*任务列表容器样式*/
08  .tasks {
09      margin: 50px auto;
10      width: 240px;
11      background: white;
12      border: 1px solid #cdd3d7;
13      border-radius: 4px;
14      -webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.05);
15      box-shadow: 0 1px 2px rgba(0, 0, 0, 0.05);
16  }
17  /*任务列表标题样式*/
18  .tasks-header {
19      position: relative;
20      line-height: 24px;
21      padding: 7px 15px;
22      color: #5d6b6c;
23      text-shadow: 0 1px rgba(255, 255, 255, 0.7);
24      background: #f0f1f2;

```

```

25     border-bottom: 1px solid #d1d1d1;
26     border-radius: 3px 3px 0 0;
27     background-image: -webkit-linear-gradient(top, #f5f7fd, #e6eaeec);
28     background-image: -moz-linear-gradient(top, #f5f7fd, #e6eaeec);
29     background-image: -o-linear-gradient(top, #f5f7fd, #e6eaeec);
30     background-image: linear-gradient(to bottom, #f5f7fd, #e6eaeec);
31     -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.5), 0 1px rgba(0, 0, 0, 0.03);
32     box-shadow: inset 0 1px rgba(255, 255, 255, 0.5), 0 1px rgba(0, 0, 0, 0.03);
33 }
34 /*任务标题样式*/
35 .tasks-title {
36     line-height: inherit;
37     font-size: 14px;
38     font-weight: bold;
39     color: inherit;
40 }
41 /*任务列表样式*/
42 .tasks-lists {
43     position: absolute;
44     top: 50%;
45     right: 10px;
46     margin-top: -11px;
47     padding: 10px 4px;
48     width: 19px;
49     height: 3px;
50     font: 0/0 serif;
51     text-shadow: none;
52     color: transparent;
53 }
54 /*任务状态样式*/
55 .tasks-lists:before {
56     content: "";
57     display: block;
58     height: 3px;
59     background: #8c959d;
60     border-radius: 1px;
61     -webkit-box-shadow: 0 6px #8c959d, 0 -6px #8c959d;
62     box-shadow: 0 6px #8c959d, 0 -6px #8c959d;
63 }
64 /*任务元素样式*/
65 .tasks-list-item {
66     display: block;
67     line-height: 24px;
68     padding: 12px 15px;
69     cursor: pointer;

```

```

70     -webkit-user-select: none;
71     -moz-user-select: none;
72     -ms-user-select: none;
73     user-select: none;
74 }
75 .tasks-list-item + .tasks-list-item {
76     border-top: 1px solid #f0f2f3;
77 }

```

HTML 结构包括任务列表标题、任务名、任务状态标识。在 CSS 代码实现的过程中，首先设计容器的样式，在代码第 7~16 行设计的容器样式中，设置宽度、背景颜色、边框及圆角，并使用 `box-shadow` 设置投影；代码第 17~33 行设置任务列表标题样式，包括设置相对定位、行高、文字投影、背景颜色及渐变；代码第 34~40 行设置任务名样式；代码第 55~63 行使用 `:before` 伪类设置任务状态样式。

11.15 微店购物车

随着移动用户的增长，基于移动端的购物服务也发展得越来越丰富，本节介绍一种微店购物车的实现方式。微店购物车包括购物车名、商品列表、总价、结算等元素。其中，商品列表包括商品图片、商品名称、商品简介、商品价格等内容，本例效果如图 11.15 所示。



图 11.15 微店购物车

实例 HTML 代码如下：


```

01 <!--购物车容器-->
02 <div class="cart">
03     <div class="cart-top">
04         <!--购物车标题-->
05         <h2 class="cart-top-title">购物车</h2>
06         <div class="cart-top-info">已选 3 件</div>
07     </div>
08     <!--购物车商品列表 -->
09     <ul>
10         <li class="cart-item">
11             <!--购物车商品详情 -->
12             <span class="cart-item-pic">
13                 
14             </span>
15             时尚耳钉
16             <span class="cart-item-desc">春装</span>
17             <span class="cart-item-price">¥ 16.80</span>
18         </li>
19         <li class="cart-item">
20             <span class="cart-item-pic">
21                 
22             </span>
23             美的空调
24             <span class="cart-item-desc">美的物联网</span>
25             <span class="cart-item-price">¥ 15.00</span>
26         </li>
27         <li class="cart-item">
28             <span class="cart-item-pic">
29                 
30             </span>
31             盆景
32             <span class="cart-item-desc">传奇神话麦迪</span>
33             <span class="cart-item-price">¥ 33.00</span>
34         </li>
35     </ul>
36     <!--购物车总计价格及结算按钮 -->
37     <div class="cart-bottom">
38         共计: ¥ 64.80
39         <a href="#" class="cart-button">结算</a>
40     </div>
41 </div>

```

CSS 代码如下:

```

01 /*购物车容器样式*/
02 .cart {

```

```
03     margin: 50px auto 0;
04     width: 300px;
05     overflow: hidden;
06     color: white;
07     text-shadow: 0 1px rgba(0, 0, 0, 0.6);
08     background: #525252;
09     border: 1px solid #202020;
10     border-radius: 3px;
11     -webkit-box-shadow: 0 1px 5px rgba(0, 0, 0, 0.5);
12     box-shadow: 0 1px 5px rgba(0, 0, 0, 0.5);
13 }
14 /*购物车标题栏样式*/
15 .cart-top {
16     position: relative;
17     z-index: 1;
18     height: 24px;
19     line-height: 24px;
20     padding: 8px 15px;
21     font-size: 14px;
22     font-weight: bold;
23     color: #eee;
24     text-shadow: 0 1px 1px rgba(0, 0, 0, 0.8);
25     background: #404040;
26     border-bottom: 1px solid #222;
27     border-radius: 2px 2px 0 0;
28     -webkit-font-smoothing: antialiased;
29     background-image: -webkit-linear-gradient(top, #525252, #3d3d3d 80%, #383838);
30     background-image: -moz-linear-gradient(top, #525252, #3d3d3d 80%, #383838);
31     background-image: -o-linear-gradient(top, #525252, #3d3d3d 80%, #383838);
32     background-image: linear-gradient(to bottom, #525252, #3d3d3d 80%, #383838);
33     -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.05), inset 0 0 0 1px rgba(255, 255, 255, 0.1), 0 1px 2px rgba(0, 0, 0, 0.25);
34     box-shadow: inset 0 1px rgba(255, 255, 255, 0.05), inset 0 0 0 1px rgba(255, 255, 255, 0.1), 0 1px 2px rgba(0, 0, 0, 0.25);
35 }
36 /*购物车物品价格样式*/
37 .cart-top-info {
38     float: right;
39 }
40 /*购物车物品列表元素样式*/
41 .cart-item {
42     position: relative;
43     line-height: 20px;
44     padding: 10px 80px 10px 15px;
45     font-weight: bold;
```

```

46     background: #525252;
47     border-bottom: 1px solid #222;
48     background-image: -webkit-linear-gradient(top, #575757, #4e4e4e);
49     background-image: -moz-linear-gradient(top, #575757, #4e4e4e);
50     background-image: -o-linear-gradient(top, #575757, #4e4e4e);
51     background-image: linear-gradient(to bottom, #575757, #4e4e4e);
52     -webkit-box-shadow: inset 0 0 0 1px rgba(255, 255, 255, 0.08);
53     box-shadow: inset 0 0 0 1px rgba(255, 255, 255, 0.08);
54 }
55 /*购物车物品 图片样式*/
56 .cart-item-pic {
57     position: relative;
58     float: left;
59     margin: -2px 12px 0 -7px;
60 }
61 /*购物车物品 价格*/
62 .cart-item-price {
63     position: absolute;
64     top: 50%;
65     right: 15px;
66     margin-top: -10px;
67     color: #eee;
68 }
69 /*购物车总计样式*/
70 .cart-bottom {
71     line-height: 31px;
72     padding: 10px 10px 10px 15px;
73     font-weight: bold;
74     background: #484848;
75     background-image: -webkit-linear-gradient(top, #545454, #434343);
76     background-image: -moz-linear-gradient(top, #545454, #434343);
77     background-image: -o-linear-gradient(top, #545454, #434343);
78     background-image: linear-gradient(to bottom, #545454, #434343);
79     -webkit-box-shadow: inset 0 0 0 1px rgba(255, 255, 255, 0.08);
80     box-shadow: inset 0 0 0 1px rgba(255, 255, 255, 0.08);
81 }
82 /*购物车按钮样式*/
83 .cart-button {
84     line-height: 29px;
85     padding: 0 25px;
86     color: white;
87     text-decoration: none;
88     text-shadow: 0 1px rgba(0, 0, 0, 0.3);
89     background: #5aa327 padding-box;
90     border: 1px solid #333;

```

```

91     border-radius: 4px;
92     background-image: -webkit-linear-gradient(top, rgba(255, 255, 255, 0.15), rgba(255, 255, 255, 0.1) 50%, transparent 50%, rgba(0, 0, 0, 0.04));
93     background-image: -moz-linear-gradient(top, rgba(255, 255, 255, 0.15), rgba(255, 255, 255, 0.1) 50%, transparent 50%, rgba(0, 0, 0, 0.04));
94     background-image: -o-linear-gradient(top, rgba(255, 255, 255, 0.15), rgba(255, 255, 255, 0.1) 50%, transparent 50%, rgba(0, 0, 0, 0.04));
95     background-image: linear-gradient(to bottom, rgba(255, 255, 255, 0.15), rgba(255, 255, 255, 0.1) 50%, transparent 50%, rgba(0, 0, 0, 0.04));
96     -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.15), inset 0 0 0 1px rgba(255, 255, 255, 0.1);
97     box-shadow: inset 0 1px rgba(255, 255, 255, 0.15), inset 0 0 0 1px rgba(255, 255, 255, 0.1);
98 }

```

代码第 1~13 设置购物车容器样式，包括宽度、容器投影、字体颜色、背景颜色等，并设置 `overflow: hidden`；代码第 14~35 行设置购物车标题栏样式；代码第 40~54 行设置购物车物品列表元素样式；代码第 61~68 行设置购物车物品图片样式；代码第 69~81 行设置购物车总计价格样式；代码第 82~98 行设置购物车按钮样式，按钮样式可以作为公共样式的一部分，以供其他部分引用，以统一全站按钮样式。

11.16 APP 导航与提醒

通常来说，一款手机 APP 离不开导航栏，导航有助于引导用户的操作，允许用户在使用过程中，在 APP 流程的不同阶段进行自由切换与跳转。与此同时，通知栏及时提醒用户相关的信息，对于用户不擅长自觉主动获取的信息采用“推”的模式，主动向用户推送消息；设置栏对用户开放了私人定制的模式，开放了个性化的接口，便于用户自定义特性字体、皮肤、快捷键等。本节介绍一种手机 APP 导航与提醒的设计与实现，效果如图 11.16 所示。

该导航的实现使用了主流的圆角效果，包括圆圈显示数字项。导航结构采用无序列表 `ul`，列表项之间使用虚线分隔开来。



图 11.16 APP 导航与提醒

实例 HTML 代码如下：

```

01 <div class="container">
02   <nav class="nav">

```

```

03     <ul class="nav-list">
04         <li><a href="index.html" class="nav-link"> 首 页 <span class="nav-counter
nav-counter- green">4</span></a></li>
05         <li><a href="index.html" class="nav-link"> 设 置 <span class="nav-counter
nav-counter- blue">8</span></a></li>
06         <li><a href="index.html" class="nav-link">通知<span class="nav-counter">15</span>
</a></li>
07         <li><a href="index.html" class="nav-link">退出</a></li>
08     </ul>
09 </nav>
10 </div>

```

CSS 代码如下:

```

01 /*容器样式*/
02 .container {
03     margin: 120px auto;
04     width: 480px;
05     text-align: center;
06 }
07 .container .nav {
08     display: inline-block;
09     text-align: left;
10 }
11 /*导航条样式*/
12 .nav {
13     padding: 4px;
14     background: rgba(0, 0, 0, 0.04);
15     border-radius: 23px;
16     -webkit-box-shadow: inset 0 1px rgba(0, 0, 0, 0.08), 0 -1px rgba(0, 0, 0, 0.3), 0 1px
17     rgba(255, 255, 255, 0.12);
18     box-shadow: inset 0 1px rgba(0, 0, 0, 0.08), 0 -1px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255,
19     255, 0.12);
20 }
21 /*导航列表样式*/
22 .nav-list {
23     padding: 0 6px;
24     height: 34px;
25     background: #f4f5f7;
26     border-radius: 18px;
27     background-image: -webkit-linear-gradient(top, white, #e1e2eb);
28     background-image: -moz-linear-gradient(top, white, #e1e2eb);
29     background-image: -o-linear-gradient(top, white, #e1e2eb);
30     background-image: linear-gradient(to bottom, white, #e1e2eb);
31     -webkit-box-shadow: inset 0 0 0 1px rgba(255, 255, 255, 0.3), 0 1px 1px rgba(0, 0, 0, 0.2);
32     box-shadow: inset 0 0 0 1px rgba(255, 255, 255, 0.3), 0 1px 1px rgba(0, 0, 0, 0.2);
33 }

```

```

32 .nav-list > li {
33     float: left;
34     height: 17px;
35     margin: 8px 0;
36 }
37 .nav-list > li + li {
38     border-left: 1px dotted #989ca8;
39 }
40 /*导航链接样式*/
41 .nav-link {
42     float: left;
43     position: relative;
44     margin-top: -8px;
45     padding: 0 14px;
46     line-height: 34px;
47     font-size: 10px;
48     font-weight: bold;
49     color: #555;
50     text-decoration: none;
51     text-shadow: 0 1px white;
52 }
53 .nav-link:hover {
54     color: #333;
55     text-decoration: underline;
56 }
57 /*提醒计数样式*/
58 .nav-counter {
59     position: absolute;
60     top: -1px;
61     right: 1px;
62     min-width: 8px;
63     height: 20px;
64     line-height: 20px;
65     margin-top: -11px;
66     padding: 0 6px;
67     font-weight: normal;
68     color: white;
69     text-align: center;
70     text-shadow: 0 1px rgba(0, 0, 0, 0.2);
71     background: #e23442;
72     border: 1px solid #911f28;
73     border-radius: 11px;
74     background-image: -webkit-linear-gradient(top, #e8616c, #dd202f);
75     background-image: -moz-linear-gradient(top, #e8616c, #dd202f);
76     background-image: -o-linear-gradient(top, #e8616c, #dd202f);

```

```

77     background-image: linear-gradient(to bottom, #e8616c, #dd202f);
78     -webkit-box-shadow: inset 0 0 1px 1px rgba(255, 255, 255, 0.1), 0 1px rgba(0, 0, 0, 0.12);
79     box-shadow: inset 0 0 1px 1px rgba(255, 255, 255, 0.1), 0 1px rgba(0, 0, 0, 0.12);
80 }
81 /*绿色计数样式*/
82 .nav-counter-green {
83     background: #75a940;
84     border: 1px solid #42582b;
85     background-image: -webkit-linear-gradient(top, #8ec15b, #689739);
86     background-image: -moz-linear-gradient(top, #8ec15b, #689739);
87     background-image: -o-linear-gradient(top, #8ec15b, #689739);
88     background-image: linear-gradient(to bottom, #8ec15b, #689739);
89 }
90 /*蓝色计数样式*/
91 .nav-counter-blue {
92     background: #3b8de2;
93     border: 1px solid #215a96;
94     background-image: -webkit-linear-gradient(top, #67a7e9, #2580df);
95     background-image: -moz-linear-gradient(top, #67a7e9, #2580df);
96     background-image: -o-linear-gradient(top, #67a7e9, #2580df);
97     background-image: linear-gradient(to bottom, #67a7e9, #2580df);
98 }

```

代码第 11~18 行设置导航条样式；代码第 20~31 行设置导航列表样式；代码第 20~31 行设置导航链接样式；代码第 57~80 行设置提醒计数样式；代码第 81~98 行设置自定义提醒计数样式。

11.17 简洁记事本

记事本类似于随手签，不仅可以记录简短事宜，又可以记录较长的文本内容。在移动 APP 上设计的记事本，不仅仅在乎文字的显示，更在乎可以随时随地、简洁直观。本节介绍一种简洁记事本的实现方法，其设计风格与真实的记事本如出一辙，设计底纹使得每行文本内容清晰可辨，效果如图 11.17 所示。



图 11.17 简洁记事本

实例 HTML 代码如下：

```

01 <section class="notepad">
02   <div class="notepad-heading">
03     <h1>Notepad</h1>
04   </div>
05   <blockquote>
06     本节介绍一种简洁记事本的实现方法。
07     <br>
08     — 超实用代码段
09   </blockquote>
10   <blockquote>
11     本节介绍一种简洁记事本的实现方法。<br>
12     — 超实用代码段
13   </blockquote>
14 </section>

```

CSS 代码如下：

```

01 body {
02   font: 12px/20px 'Lucida Grande', Verdana, sans-serif;
03   color: #404040;
04   background: #3782b0;
05 }
06 /*设置记事本样式*/
07 .notepad, .notepad:before, .notepad:after {
08   background-color: white;
09   background-image: -webkit-linear-gradient(#f6abca 1px, transparent 1px), -webkit-linear-
10     gradient(#f6abca 1px, transparent 1px), -webkit-linear-gradient(#e8e8e8 1px, transparent
11     1px);
12   background-image: -moz-linear-gradient(#f6abca 1px, transparent 1px), -moz-linear-grad
13     ient(#f6abca 1px, transparent 1px), -moz-linear-gradient(#e8e8e8 1px, transparent
14     1px);
15   background-image: -o-linear-gradient(#f6abca 1px, transparent 1px), -o-linear-gradie
16     nt(#f6abca 1px, transparent 1px), -o-linear-gradient(#e8e8e8 1px, transparent 1px);
17   background-image: linear-gradient(#f6abca 1px, transparent 1px), linear-gradient(#f6abca
18     1px, transparent 1px), linear-gradient(#e8e8e8 1px, transparent 1px);
19   background-size: 1px 1px, 1px 1px, 23px 23px;
20   background-repeat: repeat-y, repeat-y, repeat;
21   background-position: 22px 0, 24px 0, 0 50px;
22   border-radius: 2px;
23   -webkit-box-shadow: 0 0 0 1px rgba(0, 0, 0, 0.15), 0 0 4px rgba(0, 0, 0, 0.5);
24   box-shadow: 0 0 0 1px rgba(0, 0, 0, 0.15), 0 0 4px rgba(0, 0, 0, 0.5);
25 }
26 /*设置记事本容器样式*/
27 .notepad {
28   position: relative;
29   margin: 60px auto;
30   padding: 0 23px 14px 35px;

```



```

25     width: 300px;
26     line-height: 23px;
27     font-size: 11px;
28     color: #666;
29 }
30 /*设置记事本内容段落、引用样式*/
31 .notepad p, .notepad blockquote {
32     margin-bottom: 23px;
33 }
34 .notepad :last-child {
35     margin-bottom: 0;
36 }
37 .notepad:before, .notepad:after {
38     content: "";
39     position: absolute;
40     z-index: -1;
41     top: 100%;
42     left: 3px;
43     right: 3px;
44     margin-top: -2px;
45     height: 4px;
46     background-size: 1px 1px, 1px 1px, 0 0;
47 }
48 .notepad:before {
49     z-index: -2;
50     left: 6px;
51     right: 6px;
52     height: 6px;
53     background-color: #eee;
54 }
55 /*记事本标题样式*/
56 .notepad-heading {
57     position: relative;
58     margin: 0 -23px 14px -35px;
59     height: 38px;
60     background: #14466a;
61     border-radius: 2px 2px 0 0;
62     background-image: -webkit-linear-gradient(top, #226797, #0c3452);
63     background-image: -moz-linear-gradient(top, #226797, #0c3452);
64     background-image: -o-linear-gradient(top, #226797, #0c3452);
65     background-image: linear-gradient(to bottom, #226797, #0c3452);
66     -webkit-box-shadow: inset 0 1px #2f81ad, 0 2px 1px rgba(0, 0, 0, 0.4), 0 0 0 1px rgba(0, 0, 0, 0.5), 0 1px black;
67     box-shadow: inset 0 1px #2f81ad, 0 2px 1px rgba(0, 0, 0, 0.4), 0 0 0 1px rgba(0, 0, 0, 0.5), 0 1px black;
68 }

```

```

69 .notepad-heading > h1 {
70     line-height: 36px;
71     font-size: 14px;
72     color: white;
73     text-align: center;
74     text-shadow: 0 -1px rgba(0, 0, 0, 0.7);
75 }
76 .notepad-heading:before, .notepad-heading:after {
77     content: "";
78     position: absolute;
79     bottom: 2px;
80     left: 1px;
81     right: 1px;
82     height: 0;
83     border-top: 1px dashed #617c90;
84     border-color: rgba(255, 255, 255, 0.35);
85 }
86 .notepad-heading:after {
87     bottom: 3px;
88     border-color: #071c2c;
89     border-color: rgba(0, 0, 0, 0.5);
90 }

```

HTML 结构代码中使用 `section` 标签包裹记事本主体内容区域。CSS 代码第 2~14 行设置的 `.notepad` 样式即记事本容器样式，包括背景条纹样式，这是最关键的一段代码。其中，设置属性 `background-image` 为 `linear-gradient`，并分别设置加上 `-webkit-`、`-moz-`、`-o-` 前缀以及不加前缀，以兼容不同的浏览器；`background-image` 设置了 3 个 `linear-gradient` 属性，分别为记事本背景上左侧的两条竖线以及纵向重复的灰线，配合设置 `background-size`、`background-repeat`、`background-position` 属性。代码第 13 行设置 `background-size: 1px 1px, 1px 1px, 23px 23px` 即 3 组 `background-size` 大小，每组分别为 x 轴方向与 y 轴方向上与之对应的 `background-image` 的大小。代码第 14 行设置 `background-repeat: repeat-y, repeat-y, repeat`，为 3 组 `background-image` 设置 `background-repeat` 属性，分别为 y 轴重复、y 轴重复、x 轴与 y 轴重复，前两个 `background-repeat` 均为 y 轴重复且 `background-size` 均为 1px，因此呈现为两条竖线。代码第 15 行设置 `background-position: 22px 0, 24px 0, 0 50px`，为 3 组 `background-image` 设置背景位置，每组的两个值分别代表 `left`、`top` 值，即距离元素左、上的距离，第 2 条竖线的 `background-position` 的 `left` 值比第 1 条竖线的 `left` 值大 2px，因此两条竖线中间间隔 1px。纵向底纹 `linear-gradient(#e8e8e8 1px, transparent 1px)`，其中 1px 表示 `#e8e8e8` 与 `transparent` 色值只显示 1px，但 `background-size` 为 23px 23px 且 `background-repeat: repeat`，因此每隔 23 像素呈现色值为 `#e8e8e8` 的横线。

除了背景底纹设置之外，代码还通过伪类 `:before` 与 `:after` 来设置重叠效果，伪类 `:before` 与 `:after` 的背景底纹与容器样式一致，但通过代码第 37~54 行设置绝对定位 `position: absolute`，并设置 `left` 与 `right` 的值，`:before` 的 `right` 值为 6px，`:after` 的 `right` 值为 3px，恰巧呈现重叠效果，详见图 11.17 所示。

11.18 手机文件下载

随着在手机上下载文件的增多，APP 也需要下载文件的有效管理，包括下载文件列表、下载文件夹、下载进度等管理。本节介绍一种手机文件下载页面，其效果如图 11.18 所示，该文件下载界面，包括下载文件标题、文件下载列表操作区、正在下载文件简介、正在下载进度条、下载文件列表等内容。这类效果在日益流行的网盘和各种电子阅读器中使用居多。



图 11.18 手机文件下载

实例 HTML 代码如下：

```

01 <section class="widget">
02   <div class="widget-heading">
03     <h3 class="widget-title">下载文件</h3>
04     <a href="index.html" class="widget-close"><span class="icon-close">Close</span></a>
05   </div>
06   <ul class="widget-toolbar">
07     <li><a href="index.html">打开下载文件夹</a></li>
08     <li><a href="index.html">清除全部</a></li>
09   </ul>
10   <ul class="downloads">
11     <li class="download download-active">
12       <span class="icon-psd-40 download-icon"></span>
13       <h4 class="download-name">CSS 代码段.psd</h4>
14       <p class="download-info">4.8 of 15MB - 162KB/sec - 1 min</p>
15       <div class="download-progress">
16         <div class="download-progress-bar" style="width: 32%"></div>
17       </div>
18     </li>
19     <li class="download">
20       <span class="icon-ai-24 download-icon"></span>
21       <h4 class="download-name">美丽乡村.png</h4>
22     </li>
23     <li class="download">
24       <span class="icon-ai-24 download-icon"></span>
25       <h4 class="download-name">都教授.png</h4>

```

```
26     </li>
27   </ul>
28 </section>
```

CSS 代码如下:

```
01  /*下载图标*/
02  [class*="icon-"] {
03    display: block;
04    width: 16px;
05    height: 16px;
06    background-image: url("http://s3.cssflow.com/snippets/26-download-widget/img/sprite.png");
07    background-repeat: no-repeat;
08    background-color: transparent;
09    font: 0/0 serif;
10    text-shadow: none;
11    color: transparent;
12  }
13  /* 下载容器样式 */
14  .widget {
15    margin: 80px auto;
16    width: 280px;
17    overflow: hidden;
18    background: #fafafa;
19    border-radius: 4px;
20    -webkit-box-shadow: 0 0 4px rgba(0, 0, 0, 0.4);
21    box-shadow: 0 0 4px rgba(0, 0, 0, 0.4);
22  }
23  /* 下载标题样式*/
24  .widget-heading {
25    position: relative;
26    z-index: 2;
27    padding: 8px;
28    height: 24px;
29    background: #eee;
30    border-radius: 4px 4px 0 0;
31    background-image: -webkit-linear-gradient(top, #eeeeee, #d8d8d8);
32    background-image: -moz-linear-gradient(top, #eeeeee, #d8d8d8);
33    background-image: -o-linear-gradient(top, #eeeeee, #d8d8d8);
34    background-image: linear-gradient(to bottom, #eeeeee, #d8d8d8);
35    -webkit-box-shadow: inset 0 21px rgba(255, 255, 255, 0.22);
36    box-shadow: inset 0 21px rgba(255, 255, 255, 0.22);
37  }
38  /*关闭按钮样式*/
39  .widget-close {
40    float: right;
41    margin: 0 1px 0 10px;
42    padding: 3px 3px 3px 4px;
```

```

43     border: 1px solid rgba(0, 0, 0, 0.12);
44     border-radius: 2px;
45     -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.3);
46     box-shadow: inset 0 1px rgba(255, 255, 255, 0.3);
47 }
48 /*下载列表控制栏*/
49 .widget-toolbar {
50     position: relative;
51     height: 18px;
52     line-height: 18px;
53     padding: 6px 0;
54     background: #5c5c5c;
55     background-image: -webkit-linear-gradient(top, #878787, #5c5c5c);
56     background-image: -moz-linear-gradient(top, #878787, #5c5c5c);
57     background-image: -o-linear-gradient(top, #878787, #5c5c5c);
58     background-image: linear-gradient(to bottom, #878787, #5c5c5c);
59     -webkit-box-shadow: inset 0 1px rgba(0, 0, 0, 0.25), inset 0 2px rgba(255, 255, 255, 0.2), 0
60     0 2px rgba(0, 0, 0, 0.5);
61     box-shadow: inset 0 1px rgba(0, 0, 0, 0.25), inset 0 2px rgba(255, 255, 255, 0.2), 0 0 2px
62     rgba(0, 0, 0, 0.5);
63 }
64 /*下载进度条*/
65 .download-progress {
66     clear: left;
67     margin: 12px 10px 4px;
68     height: 8px;
69     padding: 3px;
70     background: #ebebeb;
71     border-radius: 7px;
72     -webkit-box-shadow: inset 0 2px 3px rgba(0, 0, 0, 0.2), 0 1px white;
73     box-shadow: inset 0 2px 3px rgba(0, 0, 0, 0.2), 0 1px white;
74 }

```

手机文件下载栏包括标题栏、取消按钮、下载文件控制栏、下载文件标题栏、文件列表、进度条等内容。代码第 1~12 行设置下载图标样式；代码第 14~22 行设置下载容器样式；代码第 24~37 行设置下载标题样式；代码第 38~47 行设置关闭按钮样式；代码第 49~60 行设置下载列表控制栏样式；代码第 62~72 行设置下载进度条样式。

11.19 迷你下拉列表框

较多的 APP 都包含设置项，方便用户对 APP 进行个性化设置、更改账号入口等。在设置中，我们常常通过下拉列表把诸多设置项的具体内容集中显示出来。本节介绍一种手机迷你下拉列表框的设计与实现，其效果如图 11.19 所示。



图 11.19 迷你下拉列表框

实例 HTML 代码如下：

```
01 <section class="container">
02   <div class="dropdown">
03     <a href="index.html">设置</a>
04     <div>
05       <ul>
06         <li><a href="index.html">账号管理</a></li>
07         <li><a href="index.html">账单设置</a></li>
08         <li><a href="index.html">通知</a></li>
09       </ul>
10     </div>
11   </div>
12 </section>
```

CSS 代码如下：

```
01 /*容器样式*/
02 .container {
03   margin: 50px auto;
04   width: 640px;
05   text-align: center;
06 }
07 /*下拉框*/
08 .dropdown {
09   position: relative;
10   height: 27px;
11   background: #363d47;
12   border: 1px solid;
13   border-color: #272c33 #242930 #22272d;
14   border-radius: 3px;
15   display: inline-block;
16   vertical-align: baseline;
17   zoom: 1;
18   *display: inline;
```

```

19     *vertical-align: auto;
20     background-image: -webkit-linear-gradient(top, #4a5361, #363d47);
21     background-image: -moz-linear-gradient(top, #4a5361, #363d47);
22     background-image: -o-linear-gradient(top, #4a5361, #363d47);
23     background-image: linear-gradient(to bottom, #4a5361, #363d47);
24     -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.1), 0 1px 2px rgba(0, 0, 0, 0.15);
25     box-shadow: inset 0 1px rgba(255, 255, 255, 0.1), 0 1px 2px rgba(0, 0, 0, 0.15);
26 }
27 /*下拉框容器*/
28 .dropdown div {
29     float: left;
30     height: 27px;
31     width: 26px;
32     background: url("../img/toggle.png") 9px 9px no-repeat;
33     border-left: 1px solid #292e36;
34     border-color: rgba(0, 0, 0, 0.4);
35     -webkit-box-shadow: inset 1px 0 rgba(255, 255, 255, 0.08);
36     box-shadow: inset 1px 0 rgba(255, 255, 255, 0.08);
37 }
38 /*下拉列表 hover 特效*/
39 .dropdown div:hover {
40     -webkit-box-shadow: inset 1px 0 rgba(255, 255, 255, 0.04), inset 26px 26px rgba(0, 0, 0, 0.1);
41     box-shadow: inset 1px 0 rgba(255, 255, 255, 0.04), inset 26px 26px rgba(0, 0, 0, 0.1);
42 }

```

代码第 1~7 行设置容器样式；代码第 8~26 行设置下拉框样式；代码第 27~37 行设置下拉框容器样式；代码第 38~42 行为下拉列表设置 hover 特效。

11.20 Google Font API

在博客的排版中，有效的字体设置是最有价值的。无论是丰富多彩的博客设计，还是很棒的网络杂志，最重要的是内容与文本的可读性，这也是好的字体设计的原因。本节不再介绍排版技巧，而是介绍一种全新的概念，即 Google Fonts API，旨在帮助所有网站设计与开发者设计更加精良的、可读性更强的网站。

代码如下：

```

//CSS 代码
01  /*引入文字*/
02  <link href='http://fonts.googleapis.com/css?family=Lobster' rel='stylesheet' type='text/css'>
03  #content {
04      width: 1000px;

```

```
05     margin: auto;
06     margin-bottom: 10px;
07 }
08 #content h1 {
09     text-align: center;
10     font-size: 4em;
11     color: #111;
12     text-shadow: 0px 2px 3px #555;
13     /*设置字体*/
14     font-family: 'Lobster', arial, serif;
15 }
//HTML 代码
16 <div id="container">
17     <div id="content">
18         <h1>Google Fonts API</h1>
19     </div>
20 </div>
```

首先使用外联样式 link 引入字体，然后在需要设置该字体的元素上设置 font-family 即可。最终展示效果如图 11.20 所示。



图 11.20 Google Font API

目前，当在一个网页使用一种特定的字体时，每次请求页面时都会下载字体。如果用户在访问页面之前并没有任何缓存的字体，浏览器会自动进行字体下载，这需要稍微消耗一些用户响应时间。

使用开放的 Google Fonts API 的优点包括：

- 庞大的可供选择字体库
- 所有的带宽负载均在于 Google
- Google 响应速度极快的
- 可以使用热链接字体文件，以便有更少的文件和带宽的需求

11.21 动态提示框

特定位置的提示框实现的是当鼠标停在一个特定位置时弹出一个提示信息框，以便提示用户详细内容。对于提示信息，往往是不希望直接与网页中的主要内容混在一起的，毕竟其作用仅仅是引导用户操作。应用提示框，既可以把提示信息与页面内容分开展示，又可以使页面变得更炫、更人性化。本例展示效果如图 11.21 所示。



图 11.21 动态提示框

本节用纯 CSS 实现图 11.21 的动态提示框，代码如下：

```

01  /*提示框进行相对定位*/
02  .tooltip-container {
03      position:relative;
04      cursor:help;
05  }
06  /*提示框样式*/
07  .tooltip {
08      display:block;
09      position:absolute;
10      width:150px;
11      padding:5px 15px;
12      left:50%;
13      bottom:25px;
14      margin-left:-95px;
15      /* 提示框样式 */
16      color:#fff;
17      border:2px solid rgba(34,34,34,0.9);
18      background:rgba(51,51,51,0.9);
19      text-align:center;
20      border-radius:3px;
21      /* 提示框渐隐效果 */
22      opacity:0;
23      box-shadow:0px 0px 3px rgba(0, 0, 0, 0.3);
24      -webkit-transition:all 0.2s ease-in-out;
25      -moz-transition:all 0.2s ease-in-out;
26      -o-transition:all 0.2s ease-in-out;
27      -ms-transition:all 0.2s ease-in-out;
28      transition:all 0.2s ease-in-out;
29      -webkit-transform:scale(0);
30      -moz-transform:scale(0);
31      -o-transform:scale(0);
32      -ms-transform:scale(0);
33      transform:scale(0);
34      /* 提示框继承父容器字体样式 */

```

```

35     font-size:14px;
36     font-weight:normal;
37     font-style:normal;
38 }
39 /*实用伪类设计三角形箭头*/
40 .tooltip:before, .tooltip:after{
41     content:"";
42     position:absolute;
43     bottom:-13px;
44     left:50%;
45     margin-left:-9px;
46     width:0;
47     height:0;
48     border-left:10px solid transparent;
49     border-right:10px solid transparent;
50     border-top:10px solid rgba(0,0,0,0.1);
51 }
52 .tooltip:after{
53     bottom:-12px;
54     margin-left:-10px;
55     border-top:10px solid rgba(34,34,34,0.9);
56 }
57 /*提示框动画效果，设置透明度与缩放效果*/
58 .tooltip-container:hover .tooltip, a:hover .tooltip {
59     opacity:0.9;
60     -webkit-transform:scale(1);
61     -moz-transform:scale(1);
62     -o-transform:scale(1);
63     -ms-transform:scale(1);
64     transform:scale(1);
65 }
66 /* 自定义提示框样式 */
67 .tooltip-style1 {
68     color:#000;
69     border:2px solid #fff;
70     background:rgba(246,246,246,0.9);
71     font-style:italic;
72 }
73 .tooltip-style1:after{
74     border-top:10px solid #fff;
75 }
//HTML 代码
76 <div class="content">
77     <i class="tooltip-container"><b>超级实用 CSS 代码 1</b><span class="tooltip">Hi, 第
    一个提示框</span></i>

```

```

78      <em class="tooltip-container"><b> 超 级 实 用  C S S  代 码  2</b><span class="tooltip
      tooltip-style1"> 第二个提示框</span></em>
79    </div>

```

动态提示框的重点在于 hover 特效样式的实现，代码第 57~65 行设置提示框动画效果，设置透明度与缩放效果。代码第 66~75 行设置自定义提示框样式。

11.22 用 CSS 创建内容幻灯片

通常情况下幻灯片的实现都需要借助 JavaScript 代码（或 jQuery 框架）来实现跑马灯的效果，如本书的第 2.15 节。本节介绍一种纯 CSS 实现的幻灯片，效果如图 11.22 所示。



图 11.22 内容幻灯片

实例 HTML 代码如下：

```

01    <div id="slider">
02      <!-- :target hooks -->
03      <div id="a1">
04        <div id="a2">
05          <div id="a3">
06            <div id="a4">
07              <!-- 1 -->
08              <a href="#a2" class="up2 default">2</a>
09              <a href="#a2" class="up2">2</a>
10              <!-- 2 -->
11              <a href="#a1" class="down1">1</a>
12              <a href="#a3" class="up3">3</a>
13              <!-- 3 -->
14              <a href="#a2" class="down2">2</a>
15              <a href="#a4" class="up4">4</a>

```

```

16      <!-- 4 -->
17      <a href="#a3" class="down3">3</a>
18      <ul>
19          <li>
20              <p>
21                  通常情况下幻灯片的实现都需要借助 JavaScript 代码来
实现跑马灯的效果。本节介绍一种纯 CSS 实现的幻灯片的设计与实现方式。通常情况下幻灯
片的实现都需要借助 JavaScript 代码来实现跑马灯的效果。本节介绍一种纯 CSS 实现的幻灯
片的设计与实现方式。
22              </p>
23          </li>
24          <li>
25              <p>
26                  通常情况下幻灯片的实现都需要借助 JavaScript 代码来
实现跑马灯的效果。
                .....//省略部分文字
27              </p>
28          </li>
29          <li>
30              <p>
31                  通常情况下幻灯片的实现都需要借助 JavaScript 代码来
实现跑马灯的效果。
                .....//省略部分文字
32              </p>
33          </li>
34          <li>
35              <p>
36                  通常情况下幻灯片的实现都需要借助 JavaScript 代码来
实现跑马灯的效果。
                .....//省略部分文字
37              </p>
38          </li>
39      </ul>
40  </div>
41  <!-- End #a4 -->
42 </div>
43  <!-- End #a3 -->
44 </div>
45  <!-- End #a2 -->
46 </div>
47  <!-- End #a1 -->
48 </div>
49 <!-- End #slider -->

```

CSS 代码如下:

```

01  /*段落首行样式*/
02  p:first-line {
03      color: #5b6f7b;

```

```

04     font-family: 'Trebuchet MS',Arial,Helvetica,sans-serif;
05     font-size: 18px;
06     font-weight: bold;
07     line-height: 45px;
08     text-transform: capitalize;
09 }
10
11 p:after {
12     color: #5b6f7b;
13     font-weight: bold;
14     padding-left: 3px;
15     text-decoration: underline;
16     content: "Read More >>";
17 }
18 /*设置幻灯片高度*/
19 #slider, ul, li {
20     height: 300px;
21 }
22 /*幻灯片类别宽度与定位样式*/
23 #slider, li {
24     position: relative;
25     width: 500px;
26 }
27 /*幻灯片容器样式*/
28 #slider {
29     margin: auto;
30     overflow: hidden;
31     background: #f6f7f8;
32     box-shadow: 4px 4px 15px #aaa;
33     -o-box-shadow: 4px 4px 15px #aaa;
34     -icab-box-shadow: 4px 4px 15px #aaa;
35     -khtml-box-shadow: 4px 4px 15px #aaa;
36     -moz-box-shadow: 4px 4px 15px #aaa;
37     -webkit-box-shadow: 4px 4px 15px #aaa;
38     border: 1px solid #bcc5cb;
39     border-width: 1px 2px 2px 1px;
40     border-radius: 10px;
41     -o-border-radius: 10px;
42     -icab-border-radius: 10px;
43     -khtml-border-radius: 10px;
44     -moz-border-radius: 10px;
45     -webkit-border-radius: 10px;
46     padding: 10px;
47 }
48 /*幻灯片无序列表样式，设置绝对定位*/

```

```

49  ul {
50      list-style-type: none;
51      position: absolute;
52      left: 0px;
53      top: 0px;
54      width: 2000px;
55      -webkit-transition: left .3s linear;
56  }
57  /*幻灯片上一张、下一张定位样式*/
58  #a1:target ul {
59      left: 0px;
60  }
61  #a2:target ul {
62      left: -500px;
63  }
64  #a3:target ul {
65      left: -1000px;
66  }
67  #a4:target ul {
68      left: -1500px;
69  }
70  /*幻灯片列表浮动*/
71  li {
72      float: left;
73  }
74  /*设置上一张、下一张样式*/
75  a {
76      background-image:
77      url("http://gtms03.alicdn.com/tps/i3/T1hMPZFvXdXXXBANEj-60-128.png");
78      background-repeat: no-repeat;
79      cursor: pointer;
80      display: none;
81      height: 64px;
82      width: 30px;
83      outline-width: 0px;
84      position: absolute;
85      top: 90px;
86      text-indent: -9999px;
87      z-index: 2;
88  }

```

代码第 1~10 行设置段落首行样式；代码第 20 行设置幻灯片高度；代码第 28~47 行设置幻灯片容器样式；代码第 49~56 行设置幻灯片无序列表样式，设置绝对定位；代码第 57~73 行设置幻灯片上一张、下一张定位样；代码第 72 行设置幻灯片列表浮动；代码第 74~87 行设置上一张、下一张的样式。

11.23 打印自动显示超链接 URL

众所周知，在浏览网页时，用户不必在意超链接的真正 URL 地址，只需单击链接就可跳转到另一个网页上，但是在打印页面时，超链接只是文字，不能单击，这时就有必要在原本超链接的文字后面自动附带超链接指向的网址，这样用户便可看到链接的真实地址。图 11.23 是使用浏览器浏览的效果图，图 11.24 是打印预览图，可以看到超链接后面自动加了网址。

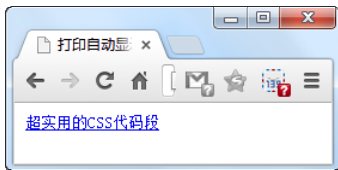


图 11.23 正常浏览



图 11.24 打印预览

上述效果只需一行代码，却节省了很多的时间：

```
@media print { a:after {content: "[" attr(href) ""]; }}
```

@media print 的意思为匹配打印设备，括号中的代码选中了所有 a 标签，在所有的 a 标签之后使用将链接的地址属性值写入到伪对象的 content 域中，于是在打印时便能在超链接之后自动添加网址。

11.24 禁用 Webkit 内核某些属性

Webkit 内核的浏览器拥有一些特殊的属性，有些属性并不常用也并不为人所知。有些时候需要禁用某些 Webkit 内核的私有属性，这在移动开发中会非常有用。

11.24.1 禁用电话号码转换为链接样式（移动设备）

iOS 系统浏览器会把纯文字转换为链接样式，当点击时与默认的系统拨号相连，但开发者并不希望这样，这种情况需要在 head 中添加如下代码：

```
<meta name="format-detection" content="telephone=no">
```

11.24.2 禁用原生弹出菜单（移动设备）

在移动设备上，如果用户长按 a 标签，将会弹出浏览器的原生菜单，添加如下的 CSS

代码可以阻止其弹出：

```
body {
    -webkit-touch-callout: none;
}
```

JavaScript 的设置方法：

```
document.documentElement.style.webkitTouchCallout = 'none';
```

注意：代码为全局设置，如果只是针对某一块元素设置，则将其写在对应的块中。

11.24.3 禁用用户选中

在有些网页中，当使用鼠标选中或是划屏选中时，网页中会出现蓝色块，影响网页的效果，这将非常不愉快。解决代码是：

```
body {
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
} /*禁用选中*/
```

使用 JavaScript 也可以，如：

```
document.documentElement.style.webkitUserSelect= 'none';
```

11.24.4 禁用输入框、文本框的轮廓线

当输入框被设置为圆角时，某些浏览器下，边框的轮廓（原始矩形）是仍然可见的。如图 11.25，在谷歌浏览器下就会出现这一问题，严重影响网页效果。使用 `outline` 属性禁用轮廓：

```
input {outline: none;}
textarea {outline: none;}
```

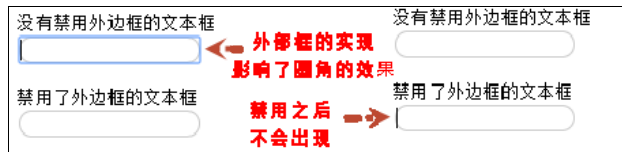


图 11.25 不禁用轮廓线与禁用轮廓线的区别

11.24.5 禁用文本框的缩放功能

文本框的缩放功能虽说是 CSS 3 中新增的用户界面属性（使用 `resize` 设置），但是在 Webkit 浏览器中设置的文本框也可以缩放，如果不希望被用户重新定义大小，可以使用 CSS：

```
textarea {resize: none;}
```